HYPERSIGNAL-MACRO USERS GUIDE

(revision 4.0)
March, 2000

Signalogic, Inc.
9617 Wendell @ Skillman
Dallas, TX 75243
tel: 214-343-0069
fax: 214-343-0163

http:// www.signalogic.com
tech support: dsphelp@signalogic.com

this page intentionally left blank

SOFTWARE LICENSE AGREEMENT

Please review the following terms and conditions carefully before using software contained in this package. **In the Agreement below, "your use of this software" means** ***use of the software on one machine at a time by an individual who has obtained the software legally, either by his/her own means or through the company or institution he/she works for.***

Pursuant to this Agreement, you may not copy any portion of the software for purposes other than creating backup copies or to support your use of the software. Such copies, modifications, or merged portions may not, in any form, be used for resale, lent to other parties, given away, or allowed to reach other parties through negligence, or carelessness. Any portion of this software merged into other software will still be subject to the terms and conditions of this Agreement, and any resulting new software must also include the copyright notice.

If it should be discovered that the individual and/or company that has obtained the program has, through resale, lending, giving away, negligence, or carelessness, enabled another party to include the program or part of the program in a product similar or competitive to "Hypersignal" series software, ***Signalogic is prepared to prosecute to the fullest extent of the law.***

You may terminate this license by destroying the program and all copies, modifications, and merged portions in any form. You may not sublicense, assign, or transfer the license or program in any form. Any such attempt is hereunder null and void.

This Agreement will be construed, interpreted, and governed by the laws of the state of Texas.

Use, duplication, or disclosure by the U.S. government of the computer software and documentation in this package shall be subject to the restricted rights under DFARS 52.227-7013 applicable to commercial computer software.

You acknowledge that you have read this agreement, understand it, and agree to be bound by its terms and conditions; you further agree that it is the complete and exclusive statement of the agreement between us which supersedes any proposal or prior agreement, oral or written, and any other communications between us relating to the subject matter of this agreement.

LIMITED WARRANTY

For a period of 90 days from the date of delivery, Signalogic, Inc. warrants to the original purchaser that the computer software described herein shall be free from defects in material and workmanship under normal use and service. During the warranty period, if a defect should occur, the product must be returned to Signalogic for repair, and proof of purchase must be presented. If product is delivered by mail, purchaser agrees to insure the product or assume the risk of loss or damage in transit, to prepay shipping charges, and to use the original shipping container or equivalent. Signalogic will, at its option, repair or replace products under warranty at no additional charge except for service to repair damage resulting from accident, disaster, misuse, abuse, or modification not authorized by Signalogic. Repair parts and replacement products will be furnished on an exchange basis and all replaced parts and products become the property of Signalogic. Signalogic, Inc., shall have no liability or responsibility to customer or any other person or entity with respect to any liability, loss or damage caused or alleged to be caused directly or indirectly by computer equipment or programs sold by Signalogic, including but not limited to any interruption of service, loss of business or anticipatory profits or consequential damages resulting from the use or operation of such computer peripherals or computer programs. Some states do not allow certain limitations to warranties so the above conditions may not apply. Therefore, although this warranty gives specific legal rights, the purchaser may have other rights which vary from state to state. ***Please fill out and return the registration form (on back of this page) to register your software purchase for warranty purposes and to ensure prompt technical support!***

REGISTRATION FORM

Name:_____

Company:_____ Dept:_____

Address: _____

City:_____ State:____Zip:_____

Telephone: (___) _____ Ext: _____

E-mail: _____

Serial No. (displayed on main menu):_____

Please fill out the above form in its entirety, photocopy or tear out this page and send to:

Signalogic, Inc. 9617 Wendell @ Skillman Dallas, TX 75243

USER FEEDBACK FORM

We at Signalogic pride ourselves on on being sensitive and responsive to customer feedback and suggestions; in fact, this is essentially how we improve existing products, and also helps us greatly to shape new products. We would appreciate any criticism, comments, and suggestions you might have. Please take the time during a slow moment in your work to complete the following form, photocopy or tear out, and return to Signalogic:

Rate installation procedure from 1 to 10:_____
Comments on installation procedure:_____

Rate user interface from 1 to 10:_____
Comments on user interface:_____

Rate documentation from 1 to 10:_____
Comments on documentation:_____

Rate overall DSPower from 1 to 10:_____
Comments on overall power:_____

Rate waveform displays from 1 to 10:_____
Comments on waveform displays:_____

Rate instrumentation displays from 1 to 10:_____
Comments on instrumentation:_____

What is the one area of Hypersignal you think needs improvement the most?_____

Name three things you tried to do with Hypersignal, but
couldn't:_____

What other software packages did you consider at the time you purchased Hypersignal?_____

What other software packages do you feel compete with Hypersignal?_____

Which DSP chip does your project use?_____

Which DSP/Acquisition hardware are you using?_____

COPYRIGHT AND TRADEMARK NOTICES

Copyright © 1985-1990, by Hyperception and 1991-2000 by Hyperception and Signalogic. All rights reserved worldwide. Reproduction or use, without permission, of editorial or illustrative content, in any manner, is prohibited. No patent liability is assumed with respect to the use of the information contained herein. While the utmost care and every precaution has been taken in the preparation of this book, Signalogic assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein. This software requires software made by Metagraphics Software Corporation to function correctly. Hypersignal software CD contain licensed program materials of Metagraphics Software Corporation, whose copyright notice is as follows:

Copyright © 1986-Metagraphics Software Corporation

Other Trademark Notices:

Compaq® is a registered trademark of Compaq Computer Corporation.

Data Translation® is a registered trademark of Data Translation, Inc.

Epson® is a registered trademark of Epson America, Inc.

Hercules™ is a trademark of Hercules Computer Technology.

HP LaserJet™ is a trademark of Hewlett-Packard Corp.

Hyperception® and Hypersignal® are registered trademarks of Hyperception, Inc.

IBM® and Proprinter® are registered trademarks, and XT™, AT™, PC-DOS™, and PS/2™ are trademarks of International Business Machines, Inc.

Intel™ is a trademark of Intel Corporation. JLaser™ is a trademark of Tall Tree Systems, Inc.

Logitech™ and Logi-Mouse™ are trademarks of Logitech, Inc.

Lotus™ is a registered trademarks of Lotus Development Corporation.

Microsoft® and MS-DOS® are registered trademarks of Microsoft, Inc.

MetaWINDOW™ is a trademark of Metagraphics Software Corporation.

Motorola® is a registered trademark of Motorola, Inc.

Mouse Systems™ is a trademark of Mouse Systems, Inc.

NEC™ is a registered trademark of Nippon Electric Corporation.

QEMM-386™ is a trademark of Quarterdeck Software, Inc.

Signalogic® and DSPower® are registered trademarks, and DirectDSP is a trademark of Signalogic, Inc.

Texas Instruments® is a registered trademark of Texas Instruments, Inc.

Turbo Pascal® is a registered trademark of Borland International, Inc.

WE® is a registered trademark, and WE® DSP32 and WE® DSP32-SL are trademarks of AT&T.

Wyse™ is a trademark of Wyse Technology.

TABLE OF CONTENTS

1.0 INTRODUCTION 1-1
1.1 Summary of Capability 1-1
1.2 Real-Time Version Capability 1-3
1.3 A Typical Hypersignal Session 1-4
1.4 System Requirements, System Options, and "Compatibility" 1-5
1.5 Manual Organization and Conventions 1-6
1.6 Hypersignal-Acoustic Documentation 1-7
1.6.1 MLS Stimulus & Response Macro Function 1-7
1.6.2 Equalize Macro Function 1-9
1.6.3 Continuous Signal Generation 1-10
1.6.4 Minphase Macro Function 1-10
1.7 Hypersignal-Marco Ex Documentation 1-11
1.7.1 Lconv and Lcorr Macro Functions 1-11
1.7.2 Lfft Macro Function 1-11
1.7.3 Continuous Wavelet Transform 1-12
1.8 Nihon Version Documentation 1-13
2.0 GETTING STARTED 2-1
2.0.1 Quick-Start 2-1
2.0.1.1 Quick-Start from CD 2-1
2.0.1.2 Quick-Start from Floppy Disk 2-1
2.0.2 What Documentation and Software You Should Have 2-2
2.0.3 Registering as a User to Ensure Technical Support 2-2
2.0.4 Making Backup Copies of Your Software 2-2
2.1 Installing Hypersignal 2-3
2.2 Running Hypersignal 2-3
2.3 Using a Math Coprocessor 2-4
2.4 Selecting Graphics and Mouse Drivers 2-5
2.4.1 Selecting a Graphics Driver 2-5
2.4.1.1 Graphics Driver Trouble-Shooting 2-6
2.4.2 Selecting a Mouse Driver 2-7
2.4.2.1 Mouse Driver Trouble-Shooting 2-7
2.4.3 Installing the Metagraphics Driver Above Conventional Memo 2-7
2.5 Basic Software Setup 2-8
2.5.1 Initializing the User Setup Area 2-8

2.5.2 Initializing the System Configuration Menu (Selecting a DSP/Acquisition Board Driver) . . . 2-8
2.5.3 Initializing the External Program Configuration Menu . . . 2-11
2.6 32-Bit Function Notes . . . 2-12
2.7 Windows Operation . . . 2-13
3.0 USING HYPERSIGNAL . . . 3-1
3.0.1 Advanced Features When Starting Hypersignal . . . 3-1
3.1 Keyboard and Mouse Interface . . . 3-3
3.2 On-Line Help . . . 3-5
3.3 The Hypersignal Main Menu . . . 3-5
3.3.1 Menu Selection . . . 3-6
3.3.2 To “Quit” Or To “Exit” Hypersignal . . . 3-6
3.3.3 The User Setup Area . . . 3-7
3.3.3.1 Dual Autoscaling . . . 3-7
3.3.4 The System Status Area . . . 3-7
3.3.4.1 Status Message Display . . . 3-8
3.3.4.2 Critical Error Message Display . . . 3-8
3.4 Using the Hypersignal Data-Entry Editor . . . 3-8
3.4.1 Waveform Summary Display . . . 3-10
3.4.2 Pop-Up Directory Display . . . 3-11
3.4.2.1 Pop-Up Directory Waveform File Search Strategy . . . 3-11
3.5 Printing Menus and Waveform Displays . . . 3-11
3.6 Invoking External Programs and the DOS “Hot-Key” . . . 3-12
3.7 Updating and Accessing the User-Environment Configuration File . . . 3-13
3.8 Creating Hypersignal Waveform Files . . . 3-14
3.8.1 Creating Complex Waveform Files . . . 3-15
3.8.2 Creating Multichannel Waveform Files . . . 3-15
3.8.3 Creating Frequency Domain Files . . . 3-15
3.9 Finding Waveforms . . . 3-15
3.10 Invoking Macros . . . 3-17
4.0 USING HYPERSIGNAL FUNCTIONS . . . 4-1
4.1 Getting Waveforms In and Out of Hypersignal . . . 4-1
4.1.1 Acquiring Waveforms From Disk Files . . . 4-1
4.1.1.1 Acquiring DSP-Chip Manufacturer Development Tool Files . . . 4-5
4.1.2 Acquiring Waveforms From DSP/Acquisition Hardware (Real-Time Software Only) . . . 4-6

4.1.3 Sending Waveforms To DSP/Acquisition Hardware (Real-Time Software Only) . . . 4-12
4.1.4 Using the ASCII Export/Import Function . . . 4-12
4.2 Displaying Time Domain Waveforms . . . 4-14
4.2.1 Time Domain Display Amplitude Scaling . . . 4-16
4.3 Using Interactive Waveform Displays . . . 4-17
4.3.1 Interactive Waveform Display Commands . . . 4-17
4.3.2 Printing Waveform Displays . . . 4-25
4.3.2.1 Using the Pop-Up Print Menu . . . 4-26
4.3.3 D/A Playback From Waveform Displays (Real-Time Software Only) . . . 4-28
4.3.4 Editing Waveforms . . . 4-28
4.3.4.1 Cut-and-Paste Waveform Editing . . . 4-28
4.3.4.2 Rubberband Waveform Editing. . . . 4-29
4.3.5 Writing Sections of Waveforms To a New File . . . 4-30
4.4 Frequency Transformation (FFT Generation) . . . 4-30
4.4.1 Framesize and Overlap Discussion . . . 4-32
4.4.2 Zero-Filling and Truncation Discussion . . . 4-33
4.4.3 Estimating Power Spectra . . . 4-33
4.4.4 LPC Autocorrelation. . . . 4-34
4.4.5 Inverse FFT Generation . . . 4-37
4.5 Displaying Frequency Domain Waveforms . . . 4-38
4.5.1 2-D Spectrograph Display . . . 4-39
4.5.2 3-D Spectrogram Display . . . 4-42
4.5.3 Magnitude Display . . . 4-44
4.5.3.1 Linear vs. Log Frequency Domain Display . . . 4-47
4.5.3.2 Frequency Domain Display Scaling . . . 4-48
4.5.4 Wrapped, Unwrapped, and Group Delay Phase Display . . . 4-50
4.5.4.1 Suggestions for Unwrapped Phase Display. . . . 4-52
4.6 Waveform Generation and Mathematics . . . 4-52
4.6.1 Writing Difference Equations . . . 4-53
4.6.2 Function Generation with Difference Equations . . . 4-59
4.6.3 Operating On Existing Waveforms with Difference Equations . . . 4-59
4.7 Using DSP/Acquisition Hardware with Hypersignal (Real-Time Software Only) . . . 4-60
4.7.1 Making Analog Measurements . . . 4-62
4.7.2 Using Instrumentation Displays . . . 4-63

4.7.2.1 Instrumentation Input 4-64
4.7.2.2 Spectrum Analyzer Data Formats 4-65
4.7.2.3 Digital Oscilloscope Triggering 4-67
4.7.2.4 Instrumentation Analog Sampling Frequency 4-68
4.7.2.5 Instrumentation Input Range and Gain 4-68
4.7.2.6 Instrumentation Display Options 4-69
4.7.2.7 Spectrum Analyzer Frequency Axis 4-69
4.7.2.8 Spectrum Analyzer Fourier Parameters 4-70
4.7.2.9 Spectrum Analyzer 3-D Map Mode 4-71
4.7.2.10 Spectrum Analyzer Amplitude Display and Calibration 4-72
4.7.2.11 Summary of Interactive Instrumentation Display Commands 4-73
4.7.3 Using Digital Record/Capture 4-77
4.7.4 Using “Stimulus & Response” 4-77
4.7.4.1 Transfer Function Analysis 4-79
4.7.5 Using a “DSP Accelerator” 4-81
4.8 Designing Filters In Hypersignal 4-82
4.8.1 Designing Classical Filters In Hypersignal 4-83
4.8.1.1 Using the Filter Construction Data-Entry Menus 4-84
4.8.1.2 Designing Lowpass Filters 4-87
4.8.1.3 Designing Highpass Filters 4-88
4.8.1.4 Designing Bandpass Filters 4-88
4.8.1.5 Designing Bandstop Filters 4-88
4.8.1.6 Designing Differentiation Filters 4-88
4.8.1.7 Designing Hilbert Transform Filters 4-89
4.8.2 Designing Arbitrary Filters In Hypersignal 4-89
4.8.2.1 Using the Graphical Filter Editor 4-90
4.8.3 Specifying Filters Outside of Hypersignal 4-94
4.8.3.1 Filter Coefficient File Format 4-94
4.8.3.2 Filter Specification File Format 4-100
4.9 Applying Filters In Hypersignal 4-101
4.9.1 Applying Filters with Recursive Filtering 4-102
4.9.2 Applying Filters with Convolution 4-103
4.9.3 Using “Snap-In” Real-Time Filtering (Real-Time Software Only) 4-105
4.9.4 Using Pole-Zero Analysis of IIR Filters 4-107
4.9.5 Generating DSP Code for Filters 4-108

4.9.5.1 Simulating the Effects of Quantization 4-114
4.10 Creating Macros. 4-116
4.10.1 Writing Macros. 4-116
4.10.1.1 Reusable Macros and Reduced Architecture 4-116
4.10.1.2 Parameter Passing From the Macro Manager Menu 4-117
4.10.1.3 Macro Procedures and “Include Files” 4-117
4.10.2 Using the Record Mode to Create Macros 4-118
5.0 HYPERSIGNAL WAVEFORM FILE FORMAT 5-1
5.1 Using the “Headitor” External Program 5-6
6.0 BRIEF DISCUSSION OF SAMPLED DATA SYSTEMS. 6-1
7.0 THE HYPERSIGNAL MACRO LANGUAGE 7-1
7.1 Macro Source, Object, and Configuration Files 7-1
7.2 Macro Language Conventions and Syntax 7-2
7.2.1 Macro Comments and Blank Lines 7-2
7.2.2 Macro Case Sensitivity 7-2
7.2.3 Macro Variables 7-2
7.2.4 Macro Constants 7-3
7.3 Invoking Hypersignal Menu Functions From Macros 7-5
7.3.1 Entering Menu Fields From Macros. 7-6
7.3.1.1 Use of the ‘#’ and ‘@’ Functions in Data Fields 7-7
7.3.2 Issuing Interactive Commands From Macros 7-9
7.4 Macro Input Parameters 7-15
7.5 Macro Assignment Statements 7-15
7.5.1 Macro Operators 7-16
7.6 Macro Loops 7-17
7.6.1 Macro Relational Comparisons 7-18
7.7 Macro If-Then-Else Constructs 7-19
7.8 Macro Procedures 7-19
7.9 Macro Goto Statements 7-20
7.10 Macro Commands and Functions 7-21
7.10.1 Macro Special Functions: ‘#’ and ‘@’ 7-30
7.11 Macro Compiler Commands 7-31
7.11.1 Macro Include Files 7-31
7.11.2 Create Text File 7-31
7.12 Compiling Macros 7-31
7.13 Executing Macros 7-31

7.14 Executing Macros and Trouble-Shooting 7-32
APPENDIX A: Macro Error Messages A-1
APPENDIX B: List of Filename Extensions B-1

1.0 INTRODUCTION

The Hypersignal-Macro Non-Real-Time (NRT) and Hypersignal-Macro software packages create and maintain an integrated digital signal processing (DSP) environment. This environment offers both an efficient menu-driven user-interface and flexible macro programming language interface. Hypersignal-Macro capability includes powerful digital signal processing functions, algorithm developmnet, simulation, analysis, and debug facilities, and several interactive waveform displays. In the case of Hypersignal-Macro (or other versions of) real-time software, real-time signal acquisition, instrumentation, file-based simulation "DSP accelerators", and real-time algorithm development are also included.

Inside this integrated environment, the Hypersignal user can, for example, guide sampled data from initial acquisition and display in the time domain, through built-in operations such as autocorrelation, digital filtering, convolution, signal arithmetic, difference equations, and power spectra generation, and on to transformation and display in the frequency domain. Using the macro language capability, the user can essentially create completely new DSP functions, based on the existing building blocks. Without the aid of Hypersignal-Macro, such analysis might involve separate program modules, display modules, and time-consuming user-written interfaces and libraries, not to mention accounting of the intermediate data files generated.

In its menu-driven mode, Hypersignal-Macro embodies a free-form "lab bench" philosophy rather than the "batch mode" philosophy of software packages that aim for total flexibility but require any and all action to be pre-programmed. Over time, individual Hypersignal menu functions have become mature, robust, and powerful—especially the interactive waveform display and editing functions, which automatically adjust to the data being displayed. Since these functions exist in an overall environment that is integrated, Hypersignal has become well-known as a package that encourages free-form thought and allows the user to focus on the problem. With the new macro language mode, Hypersignal-Macro now also allows the "pre-program" mode of operation—but the beauty is that the two modes can be intermixed freely, depending on the application and the circumstances. And the macro language is straightforward yet powerful; included are programming constructs such as if-then-else, loops, variables and their mathematical manipulation, procedures, parameter passing both from outside the macro language and inside the language (to the procedures), and, of course, remote control over all standard Hypersignal functions and interactive displays. Already, several new DSP functions have been added to Hypersignal in the form of ready-to-execute macros included as part of the distribution software.

1.1 Summary of Capability

Hypersignal provides a complete "signal processing environment", which not only allows algorithm development, but also other phases of typical signal processing system design, such as DSP chip source code development and debug. A variety of engineering and scientific problems involve sampled data systems and require digital signal processing techniques in some form for effective solution. With the advent of fast and inexpensive digital signal processors, demand has increased for software development tools that facilitate analysis and design of such systems. Hypersignal-Macro is just such a tool; by taking advantage of the flexibility and convenience of personal workstation computers, the signal processing professional can quickly analyze sampled data without undue attention to low-level detail. Some of the functions available using either Hypersignal-Macro NRT (Non-Real-Time) or Hypersignal-Macro are listed below.

Time Domain Display and Waveform Editing

Includes dual-display, variable framesize, pan, zoom, variable amplitude display scaling, random frame and/or time value access, and D/A output of target sections of displayed waveform data (real-time versions of software only). Waveform editing includes a cut-and-paste mode with copy, move, delete, and insert functions, and a rubberband mode with free-form waveform shaping, and automatic creation and maintenance of waveform backup (".BAK") files.

Frequency Domain Display

Includes spectra display (linear, log, amplitude-squared, dual-display, pan, zoom, random frame access); phase display (wrapped, unwrapped, and group delay, dual-display, pan, zoom, random frame access); 3-dimensional spectrogram (hidden-line removal, optional mesh dual-display, pan, zoom, random access); and 2-dimensional spectrograph (selectable contour, variable dB step, dual-display, linear, log, zoom, pan, random time value access, and D/A output of displayed waveform data (real-time software versions only).

Fast Fourier Transform (FFT)

Includes control over windowing, amount of frame overlap, FFT size, time domain framesize, real/imaginary output and numerical precision. In the case of real-time software versions that contain DSP board device drivers, this function can be accelerated using the currently installed DSP board.

Difference Equations

Generation of new waveforms and manipulation of existing waveforms using various operators and functions can be specified in difference equation form. Existing waveforms, unit-step and delta functions, and previous output values can be utilized as input. Operators and functions include signal arithmetic, signal calculus, trigonometric/transcendental functions, unit-step and delta functions, random noise, frame-based operations, and correlations. It is a simple matter to build sine, pulse, noise, etc. function generators, as well as to operate arbitrarily on existing waveforms.

Filter Construction

FIR design methods include Kaiser Window, Hamming Window, Hilbert Transform, differentiator, and Parks-McClellan arbitrary design. IIR designs include Butterworth, Chebyshev I & II, elliptic, Bessel, and Steiglitz arbitrary design. Arbitrary filter design includes a graphical filter editor that allows free-form or program-controlled creation and modification of new or previously edited desired frequency-domain responses. Filter editing may be performed in log, linear and zoom modes using the keyboard or a mouse—precise desired response specifications may be entered numerically via an ASCII text file. All filter designs, whether classical or arbitrary, use automatic specification and coefficient save files, for easy future reference or modification. Filter coefficients may be entered directly into ASCII files used by other program functions such as pole-zero display and automatic source code generation; coefficient entry may be in factored form with factors of varying order or in fully expanded form.

Recursive Filtering

Allows an IIR filter to be applied to an input time waveform. Input is taken from IIR filter coefficient files, which are normally generated automatically by the program, but may also be user-created.

Inverse Fast Fourier Transform (iFFT)

Using real and imaginary data files as input, frequency waveform files undergo an inverse Fast Fourier Transform to create an output time domain waveform.

Convolution/Correlation

Allows two time waveforms to be convolved or correlated with one another. This feature is intended mainly to apply FIR Filter Construction output filters on waveform data, but can be used for general convolution of any two time waveforms. In the case of real-time software versions that contain DSP board device drivers, this function can be accelerated using the currently installed DSP board.

LPC Autocorrelation

Coefficient generation is linear-predictive and output coefficients may be specified to be reflection or analysis-filter coefficients. Filter coefficients are time domain file-compatible with other time domain functions, including FFT Generation. Includes control over LPC order, time domain analysis framesize, frame overlap, and time domain window (Hamming, Bartlett, etc.). This function provides a commonly used analysis tool for speech and other semi-periodic data to extract the major frequencies (sometimes called “formants”).

Power Spectra Generation

Using previously created frequency waveform files as input, this function generates a power spectra waveform file as output. Practical methods used for averaging include sectional averaging, running average, and exponential filter.

Transfer Function Generation

Using previously created real and imaginary frequency waveform files as input, this function calculates complex ratio transfer function output, including magnitude, phase, and complex waveform files, that can be used for further frequency domain display and processing.

Pole-Zero Display

Allows poles and zeros of an infinite impulse response (IIR) filter to be viewed graphically in both the Z and S planes. Input is taken from IIR filter coefficient files, which are normally generated automatically by the program, but may also be user-created.

Code Generation

Filter code generation for Texas Instruments TMS320xx family (including TMS320C30), AT&T WE DSP32x, Motorola DSP5600x and DSP9600x, NEC 77220, and Analog Devices ADSP-210x. User controls include selectable IIR cascade structure, base address of filter coefficients, and for fixed-point DSPs, coefficient quantization (from 9 to 32 bits), optional simulation of quantization effects, and rounding.

File Acquisition

A general-purpose utility is available to import waveform data in various formats into the Hypersignal environment. Several binary and ASCII formats may be selected. Also, numerous other format options are available, including shifting and masking of individual samples, sample-width control (in 4-bit chunks), foreign header discarding, multiplexed acquisition of multichannel or complex data, unsigned and μ-Law/A-Law companded data controls, and mean (average) subtraction. This function has evolved tremendously over a period of years and has become quite powerful and flexible.

ASCII Export

Enables data transfer between different machines and programs other than Hypersignal, including standard text-editors. Hypersignal waveform data files, including waveform headers, can be exported and imported to and from the Hypersignal environment in three different ASCII formats: decimal, hexadecimal, and floating-point.

User-Defined Programs

External programs can be created (in any programming language) to perform application-specific functions. Execution of these programs can be performed from the Hypersignal main menu, allowing them to remain part of the integrated Hypersignal environment. Execution of external user-defined programs can be stand-alone, also.

1.2 Real-Time Version Capability

The following functions are active only in real-time versions of the software, and not in Hypersignal-Macro NRT (Non-Real-Time) or in the Nihon version of Hypersignal-Macro NRT. There is no NRT version of Hypersignal-Acoustic.

Analog Conversion

Using various acquisition hardware, data may be sampled in real-time and stored/read continuously to/from disk. Options include multichannel operation, input/output gain and offset control, multicondition triggering and level control (including external trigger), “snap-in” real-time filters, and simultaneous input/output operation (“stimulus and response”).

Digital Oscilloscope

Display combinations of real-time signals from the input channels of the analog hardware and previously stored waveform data. Parameters under user control include variable timebase, amplitude scaling, offset control, trigger configuration for both single and dual-trace display, and display format options such as time units and axes. Trace displays can be captured and sent to waveform file or printer.

Spectrum Analyzer

Allows dynamic display of frequency domain data using either analog signal acquisition hardware or previously stored waveform files as input. Combinations of inter- and intrachannel magnitude, phase, time, power spectra, or transfer function data may be displayed. Options include linear/log/inverse log display with line/point modes and background grid. Where analog hardware input is used, time domain analysis framesize, FFT order, real-time sampling frequency, and time domain window may be specified. Trace displays can be captured and sent to waveform file or printer.

Accelerated FFT, iFFT, and Convolution/Correlation

Using DSP accelerator hardware, processing speed can be increased from 10 to 40 times over the standard 80x86 and/or 80x87, depending on DSP chip speed and amount of DSP board memory available.

D/A Output from Waveform Display

Using currently selected analog hardware, selected sections of waveform data may be sent directly to D/A output from time and 2-D spectrograph displays using the mouse-controlled target window.

Real-Time Filtering

Using filter coefficient files previously created by Hypersignal Filter construction functions or user-generated outside of Hypersignal, digital filters can be "snapped-in" to instrumentation displays and continuous acquisition/generation of signals to/from disk. Instrumentation displays can support dual filters. Quantization may also be specified when using fixed-point DSPs.

Stimulus & Response

In addition to continuous signal acquisition to disk, the Analog Conversion function can also perform simultaneous output. By specifying a "stimulus" output waveform file, and an acquisition delay (positive or negative with respect to the onset of the stimulus), you can use the Hypersignal software to make delay, frequency response, transfer function, and other measurements of an unknown test system.

1.3 A Typical Hypersignal Session

The following outline illustrates how a typical signal analysis session might utilize some of Hypersignal's functions:

1) Input Signal Capture

By means of the real-time Analog Conversion utility (real-time software only) or File Acquisition utility (both Hypersignal-Macro and Hypersignal-Macro NRT), the user creates a time domain waveform file on disk consisting of digital, or "sampled-analog" time data representing a real-world signal of some kind. Each sample is the value of some physical quantity (such as voltage, mechanical force, fluid pressure, etc.) at an instant in time. A series of time samples, stored in a disk file along with parameter and attribute information stored in a "waveform header" which describes the nature of the data to the program, comprises the "time waveform file" (see chapter 6, BRIEF DISCUSSION OF SAMPLED DATA SYSTEMS, for more theory of sampled data).

2) Signal Display

The input signal is displayed in the time domain for verification and/or study. Sections of the input signal may be "zoomed in on" for display in greater detail, modified using the waveform editor, or output to the D/A converter of an analog board to drive speakers, etc.

3) Signal Processing

The input signal is filtered to diminish unwanted signal components and/or noise. The filter applied has been designed beforehand through Hypersignal menu selection, and can easily be traded for another or changed at any time. The signal is further run

through a difference equation that “mixes” it with a pre-defined function by multiplying in the time domain. Also, a user-defined algorithm equation is applied that seeks certain features in the waveform and creates another waveform for future evaluation of algorithm performance.

4) Frequency Transformation

The input signal undergoes Fast Fourier Transformation (FFT) that breaks it down into component frequencies, and generates magnitude and phase waveform files for subsequent display, and optional real and imaginary waveform files for subsequent inverse FFT operations.

5) Frequency Domain Display

The transformed input signal is presented graphically in the frequency domain using phase and magnitude displays, which allow close inspection of frequency characteristics. The transformed input signal is also displayed as a 3-D spectrogram, which returns a time-viewing aspect to the signal as the “apparent axis” in a three-dimensional representation. Slowly changing frequency characteristics of the signal can now be discerned. “Contour” analysis of the frequency data is obtained by viewing the 2-D spectrograph, where amplitude energy is represented by color or grey-level intensity over time and frequency.

5) Difference Equation Processing

The frequency domain waveform is now manipulated using the Difference Equations function. “Spectral subtraction” is performed, where frequency frames from one input waveform are subtracted from those of another to produce a third frequency domain waveform, which can they be displayed as in 5), above.

Of course, the above “typical session” is just an example. Many, many different types of analysis are possible. And, for many users, the algorithm design and simulation will be just the start, with the end-goal being real-time implementation on a specific DSP device.

1.4 System Requirements, System Options, and “Compatibility”

SYSTEM REQUIREMENTS

Minimum Memory:	384 kByte
Operating System:	DOS 2.0 or greater
Graphics:	Any of over 100 different types of hardware supported, including CGA, EGA, VGA, Hercules, AT&T, and popular 800x600 and 1024x768 formats. See section 2.4.1.

SYSTEM OPTIONS

Mouse:	Interface to “MOUSE.COM” memory-resident driver supported; some mice suported directly through serial ports. See section 2.4.2.
Math coprocessor:	80x87 math coprocessor automatically detected and utilized if available. See section 2.3.
Printer:	Screen-dump and publication-quality print to over 185 printers supported. An appropriate amount of EMS memory (usually about 1 MByte) is required in the system to allow publication-quality printing of waveform displays. See section 3.5.

COMPATIBILITY

The Hypersignal software package has been designed to run on IBM PC, XT, AT, 386, 486 and Pentium PCs with over 100 different types of graphics hardware and several different types of mice installed. Hypersignal will likely function correctly on many compatible or similar machines with compatible or similar graphics hardware. However, the word “compatible” means something different to every hardware and software manufacturer, and Signalogic makes absolutely no guarantees concerning compatibility.

1.5 Manual Organization and Conventions

This manual is basically organized as the following chapters:

Getting Started

Using Hypersignal (overall user-interface)

Using Hypersignal (specific functions)

Waveform File Format

Brief Discussion of Sampled Data Systems

Hypersignal Macro Language

The Getting Started chapter contains information on installing the Hypersignal software, as well as information concerning 80x87 math coprocessor usage, graphics and mouse hardware driver selection, DSP/Acquisition board driver selection (real-time software only), and basic software setup.

Chapter 3, Using Hypersignal, provides information from an overall user-interface perspective. Its purpose is to make you comfortable using the software. Some areas covered include: on-line Help, keyboard and mouse usage, the Hypersignal main menu, the Hypersignal data-entry editor, printing screen and waveform displays, invoking external programs and the DOS shell, updating the user-environment configuration file, and invoking macros.

Chapter 4, Using Hypersignal Functions, provides specific information about doing DSP-related things in Hypersignal, including "how-to" and "why" discussion. Some areas covered include: moving data in and out of Hypersignal, using interactive waveform displays, frequency transformation, displaying frequency domain data, writing difference equations, making analog measurements, designing and applying filters, and writing macros to automate all other Hypersignal functions.

The Waveform File Format chapter provides complete information about Hypersignal waveform organization and file headers.

The chapter on Sampled Data Systems provides a brief introduction to digital signal processing, and why people do it.

Finally, the Macro Language chapter contains detailed information about macro language programming and what it takes to automate algorithms and measurement sequences.

Also, please note that the following separate manuals contain further documentation about Hypersignal:

1) the "Hypersignal-Macro Software Reference Guide" contains a concise, brief list of all data-entry menu fields and interactive display commands for each Hypersignal function

2) the "Hypersignal-Macro Tutorial" contains several practical, step-by-step "real-world" examples of using the Hypersignal software

3) "Hypersignal-Macro Hardware Reference Guide" contains hardware-specific information about various DSP/Acquisition hardware when used with Hypersignal

NOTE: **As used in this manual, "real-time versions of Hypersignal" or "real-time versions of the software" include Hypersignal-Macro, Hypersignal-Acoustic, and the Nihon versions of both. "Non-real-time software" indicates Hypersignal-Macro Non-Real-Time (NRT) and the Nihon version of same. All of the real-time versions are supersets of Hypersignal-Macro NRT; that is, all algorithm development, analysis, file display, simulation, and utilities functions available in the non-real-time software are also available in the real-time software. Features found only in real-time versions of the software are so identified when they are discussed in this manual.**

Use of "HSxxxxx" refers to the appropriate executable and overlay filenames for various versions of Hypersignal (HSMACRO, HSACOUS, HSMACNRT, etc.). Use of "HSx" refers to the appropriate batch filename for various versions of Hypersignal (HSM, HSA, HS, etc.). Use of "HSxN" refers to the appropriate batch filename for a Nihon version.

Words or letters enclosed with single-quote marks (') are either keyboard commands or data-entry editor fields. In either case, the single-quote mark convention is intended to show something the user would enter into the program, via keyboard or mouse.

Words or letters enclosed with double-quote marks (") are menu labels or data-entry field labels. In general, the double-quote mark convention is intended to highlight any non-menu-field screen label or text that is generated by the Hypersignal software.

Menu selections are denoted with captialization and data-entry menu fields are denoted in all uppercase letters, neither with quote marks. Since they are referred to so often, the use of quote marks would be excessive.

Occasionally, hexadecimal numbers may be indicated with a trailing “H”, as in “33CH”.

Occasionally, non-data keyboard entry, such as control (‘Ctrl’) or carriage return sequences, may be indicated by a sort of short-hand, using surrounding ‘< >’ symbols, as in ‘<Ctrl-P>’, or ‘<Car Ret>’.

1.6 Hypersignal-Acoustic Documentation

The standard Hypersignal-Macro documentation encompasses the Hypersignal-Acoustic package. Below is a summary of important differences in the two packages and what part of the documentation covers these differences:

Hypersignal-Acoustic Differences

1) Additional macros provided on CD; see section 2.0.2.

2) Device drivers for Techron TEF20HI and Ariel ProPort systems; see the Hardware Reference Guide.

3) Log-squared amplitude display and editing option in time domain display; see section 4.3.1, Interactive Waveform Display Commands.

4) 1/N octave band display in Spectrum Analyzer; see section 4.5.3.1, Linear vs. Log Frequency Domain Display.

5) Several Spectrum Analyzer functions, including inverted trace output to filter design-goal file, output of real/imaginary transfer function data to waveform files, coherence function, etc; see section 4.7.2.11, Summary of Interactive Instrumentation Display Commands.

1.6.1 MLS Stimulus & Response Macro Function

The mls.mac function is include only with Hypersignal-Acoustic.The mls.mac function prompts for stimulus length, number of averages, output waveform filename, sampling frequency, a channel list, and stimulus amplitude and offset. It uses the currently installed hardware specified in the System Configuration menu ANALOG CONVERSION field, performs the stimulus & response operation, synchronously averages the results in the time domain, performs a cross-correlation operation to extract the impulse response, and then an FFT to calculate the frequency response. The macro function concludes by showing the frequency response in the Magnitude Display, with vertical scale set to +/- 50 dB.

When using this function, the stimulus length should be entered in seconds. The actual stimulus length in samples is chosen by taking the entered value, multiplying it by the entered sampling rate, and quantizing the result up to the nearest allowable MLS length. Quantization is due to the nature of the MLS process, which requires stimulus lengths to be a power of 2 minus 1; for example, 4095, 8191, 16383, 32767, and 65535. BE AWARE that some DSP hardware can achieve only certain sampling rates, so it is a good idea to know these ahead of time if you are trying to calculate stimulus lengths exactly. See the Hardware Reference Guide for information about memory and sampling rate limitations for individual hardware types.

Currently, mls.mac does not have an option to equalize out the response due solely to the measurement board being used. This option, and other “calibration mode” features, are expected to be added later.

Conventional Memory Considerations

The MLS function uses several “protected-mode” programs during processing. These programs need the dos4gw “DOS Extender” in order to run correctly. Dos4gw requires a piece of conventional memory (DOS memory less than 640k) in order to get started; after which it runs the protected-mode program in upper memory (memory above 1Mb). Depending on what drivers and resident programs are loaded in your machine (i.e. utilities specified in your config.sys and autoexec.bat files), there may not be enough conventional memory to run the MLS function fully. You may see a message such as “not enough memory” or “external program problem”. In such cases, make sure that as many drivers and resident programs are loaded into upper memory as possible. Specifically, make sure that your config.sys file contains the follwing lines near the beginning of the file:

```
device=C:\dos\himem.sys
DOS=HIGH,UMB
device=C:\dos\emm386.exe NOEMS ....
```

Note that the emm386.exe line may have other items after the NOEMS. The above lines will ensure that utilities are loaded into high memory. You should have about 550k free to run Hypersignal-Acoustic and the MLS program successfully. You can check this by using the "mem" command from the DOS prompt.

Stimulus Amplitude

Choosing the MLS stimulus amplitude can be a tricky business. Too much and there will be A/D converter saturation (clipping); too little and the system's dynamic range will be underutilized and a measurement with less than the maximum possible signal-to-noise ratio will result. To determine the right amplitude, run mls.mac once with a very short stimulus (e.g. 255 samples), and note the max amplitude returned in the Analog Conversion function; alternatively, check the max amplitude value in the waveform "mlsresp", by loading it into Waveform Display/Edit. If this value is 32767 (for 16-bit or less A/D converters), then clipping probably occurred. Lower the entry in the mls.mac data entry menu and run the function again. Resulting amplitudes in the 25,000 to 30,000 range are good. Note also that the max amplitude is not necessarily an accurate measure of the gain of the system under test. For example, the stimulus output amplitude might be 15,000, and max response amplitudes 25,000. Yet such a system might have little or no gain. This can occur due to delays that vary with frequency, causing amplitudes from different frequenices to superpose and create a larger amplitude.

DSP Memory Considerations

You must be aware of your DSP/analog I/O board memory size to make effective and reliable use of the mls.mac function. Some boards have lots of onboard DSP memory (either SRAM or DRAM) and let you run huge MLS stimulus sizes, some boards have very little memory, and most are in-between. Also, DSP5600x hardware currently supports an onchip dynamic MLS output generation for some longer MLS stimulus sizes. Because Hypersignal-Acoustic supports so many types of hardware, the mls.mac function is very general in nature. You will not see any warning in the Macro Manager menu if you enter a stimulus size greater than what can be contained in DSP board memory. However, if you run the stimulus & response function manually from Analog Conversion, by entering 'mlsresp' as the FILENAME field, a warning will appear if stimulus truncation occurs. A good rule of thumb is as follows:

Take the amount of DSP board memory available for data usage (see the Hardware Reference Guide to get this figure), subtract 24k (words) for input buffers, and divide the remainder by two. This will be approximatelythe maximum stimulus length in samples. If the amount of available data memory is less or equal to 32k, then subtract only 12k for input buffers.

Below is a short list of example boards, the amount of data memory available for use by Hypersignal when shipped with Signalogic-recommended memory amounts, and the maximum stimulus size. Memory amounts amounts are in words (natural wordlength of DSP device) and given in two values: Signalogic recommended amount that should be shipped with the board, and actual stimulus area available (excluding program area and input buffers). Memory amounts marked with an "*" can be packed double because the memory is accessible in words (16-bit values) as well as 32-bit values. Stimulus size values are in samples. Again, see the Hardware Reference Guide for precise information:

Board	Total Mem	Stimulus Mem	Max MLS Stimulus Size
Ariel DSP-32C	64k*	96k	32767
Ariel PC-56D + ProPort	64k	20k	262143++
Ariel DSP-16	64k	20k	8191
TBS Tahiti	32k	12k	262143++
CAC QUANTUMdsp	512k*	984k	262143
Innovative Integration PC31	128k	96k	32767
National Instruments AT-DSP2200	384k*	728k	262143

Signalogic Sig32C-8	128k*	192k	65535
Techron TEF20HI	48k	16k	262143++

++Indicates dynamic onchip generation of MLS output, which avoids stimulus memory limitation. Currently active only for some DSP5600x hardware and some stimulus lengths.

When the mls.mac function executes, the FRAMESIZE field of the Analog Conversion function will contain the stimulus length being used, and the framesize value of the resulting output waveform will also be set to the stimulus length. If this value exceeds the above values (for the given memory amounts), then results will be undefined!

For larger stimulus sizes, it is highly recommended that DSP hardware be obtained with at least an initial memory expansion option, and possibly the maximum amount of memory expansion allowable. In many cases, the DSP hardware manufacturer does not ship boards with much beyond 16k words of memory, for two reasons: in order to keep cost down, and because traditionally DSP programs have been known for compactness. Thus, it is important to know what is the memory amount status of your hardware.

Some supported DSP56xxx boards have only 32k x 24 of SRAM total, which Hypersignal allocates as 8k x 24 P: and 12k x 24 X: and Y: each. This is a small amount of memory, and limits MLS stimulus sizes to about 4095 samples. But, when using MLS stimulus & response with DSP56xxx boards, maximum stimulus sizes are longer--up to 262,143 because a dynamic MLS generation algorithm on the DSP56xxx device is used (note, however, that MLS lengths 8,191, 16,383, and 65,535 are not yet operational).

Another important memory consideration to be aware of when using mls.mac is memory speed. Memory that runs with wait-states can severely affect the maximum sampling rate possible when using the mls.mac function. For example, boards that povide DRAM for data buffers rather than SRAM can require as many as 4-6 wait-states to access the DRAM. DRAM is used to provide huge amounts of memory for reasonable cost, but there is always a tradeoff. Another example is the inexpensive multimedia class of hardware, which typically use slower SRAM (about 3 wait-states) in order to keep costs down. Generally, use of slow SRAM will cut maximum MLS sampling rates to about 25 kHz, and use of DRAM will cut maximum rates to about 16 kHz to 22 kHz. However, degradation depends also on the processor type and speed, which can alleviate the drop in maximum rates somewhat.

In general, if you suspect that a memory problem--either size or speed--may be affecting proper MLS operation, here is a diagnostic technique that can be used very reliably:

1) Run the mls.mac function. You may want to temporarily reduce the number of averages so that the time required to perform the diagnostic is reduced. However, you should not reduce the number of averages below 5.

2) Run FFT Generation on the waveform file ‘mlsresp’. This is not the normal output of the mls.mac function, but an intermediate output. Use a framesize of 250, an FFT order of 9, an overlap of 50%, and a Hamming window. No real/imag output is required.

3) Display the results using 2-D Spectrograph. Use ‘0,-85,A’ in the AMP SCALING field. Enter ‘0’ in the TIME SHOWN field to view the whole waveform. If you see any vertical artifacts, such as stripes or bars, then there may be a problem. Vertical features in this display indicatewideband energy, or energy across the entire spectrum, which indicates discontinuities or glitches (i.e. some type of “edge noise”) in the time domain. Further, if such artifacts appear at regular intervals, then there is probably a buffer problem related to memory speed or size. Basically, anything you can see that disturbs the uniformity of the this display, or is obviously different in one region than another, could represent a problem.

Aborting MLS.MAC

Normally, the ‘Esc’ or F3 keys abort executing macros in Hypersignal-Acoustic. But, during mls.mac processing, several external protected mode programs are executed, and pressing ‘Esc’ during these programs won’t do any good (until the program finishes). Thus, use the Esc, Ctrl-Break sequence to abort during protected mode program processing.

1.6.2 Equalize Macro Function

This function takes Spectrum Analyzer transfer function display (either analog signal or waveform file as input) and writes inverted linear magnitude output to a .FRQ file. This is done using the Spectrum Analyzer ‘X’ (inverted write) command, which is

active only in Hypersignal-Acoustic. The "zero dB" point used for inversion is the zero dB point on the transfer function display as default, or the current vertical position of the cursor, if displayed. The macro function then takes the inverted magnitude output, and the original phase file (which must be created beforehand if the input was analog), and calculates real/imag components. It then calculates an inverse FFT to create a time domain impulse response that can be used as an FIR filter to equalize the original transfer function curve. As a sanity check, the macro function calculates the magnitude response of this impulse response, and displays it, along with the orignal, non-inverted magnitude response, in the Magnitude Display function, using the overlay mode for comparison and a display range of +/- 70 dB.

It is important to note that the length of the resulting FIR filter will be equal to the FFT size in the transfer function display, regardless of the size of the time domain (measurement) window in the transfer function display. Reasonable curve-matching may occur only with FFT sizes of 512 points or more, which means that it is unlikely that supported DSP hardware can implement filters of this size in real-time inside Hypersignal, since all Hypersignal instrumentation functions implement FIR filters using standard multiply-accumlate time domain techniques. For these reasons, you will probably need the Real-Time Convolver to apply the equalization filters in real-time, using supported DSP/analog I/O hardware. When running Hypersignal-Acoustic under Windows, it is possible to set up and control two separate DSP/analog I/O boards at once, using two copies of HS-Acoustic. The main system would control the measurement/instrumentation board, and the second copy the Real-Time Convolver board. In this way, the real-time Spectrum Analyzer can be used to monitor the effects of the equalization, and change create a new equalization filter, as needed.

After running equalize.mac, it may be desirable to run the resulting impulse response waveform (.tim file) through the minphase.mac function. The minimum phase function does not change the magnitude response, but it attempts to minimize the overall group delay. This may produce a more "normal-looking" impulse response, as well as shorten the number of taps required if the impulse response is to be downloaded and run as an FIR filter. However, phase preservation from the original, unequalized response will be destroyed.

The equalize.mac function has been extensively tested with frequency domain waveform files generated by the TEFRD.EXE and SYSIDRD.EXE utilities, which convert real/imag output data files from Techron's Sound Lab software, and Ariel's SYSid software, to Hypersignal .FRQ and .PHZ formats. The tefrd.exe and sysidrd.exe utilties can be conveniently controlled using the tefrd.mac and sysidrd.mac macro functions; these two utilities are included only with Hypersignal-Acoustic.

When using a fixed-point DSP board at the same time as the equalize.mac function, it is suggested that the DSP ACCELERATOR field in the System Config menu be temporarily disabled. Even though the amount of error introduced by fixed-point FFT operations is small, it may be important in the equalization process, which is numerically sensitive.

1.6.3 Continuous Signal Generation

Signal generation looping in the Analog Conversion menu item has been enhanced to allow easier construction of a continuous function generator. By creating short signal/function waveforms with the Difference Equations function, and then using these waveforms as input to the Analog Conversion function, a continuous signal generator can be achieved when the ACQ/GEN field is set to 'GL'. Some guidelines to follow when specifying a Difference Equation:waveform lengths (as specified in the Difference Equations # OF SAMPLES fiel) should be an integer multiple of the function period; if you are in doubt about the function period, use the Waveform Display/Edit option to measure it exactly

The overall waveform length should be anywhere from a few hundred samples to less than 6000 samples; only one or just a few periods of the function waveform is not recommended, because of the overhead required by the DSP software when "wrapping around" to repeat the function. And, definitely, long waveforms are not recommended because they become split into buffer segments due to DSP board memory limitations, defeating the continuous mechanism altogether the length of the waveform, in samples (and multiplied by the number of channels), should be divisible by two.

Watch out for sampling frequency when designing the difference equation; most hardware only allows certain sampling frequencies, and if you base your calculations on a sampling frequency the hardware cannot achieve, then there is likely to be a discontinuity when the input waveform repeats.

1.6.4 Minphase Macro Function

This function takes an input time domain waveform, which should be a filter response or other impulse response, and creates a minimum phase version, also as a time domain waveform, with a user-specified output filename. Minphase.mac attempts to

minimize the overall group delay of the input impulse response, while leaving the frequency domain magnitude response unchanged.

The minimum phase version should have the same frequency domain magnitude response (you can do an FFT Generation to check this), but minimized phase response, and an impulse response with energy concentrated near time=zero. For example, if you run minphase.mac on a Hypersignal FIR filter, impulse response, you should see the resulting time domain impulse response looking somewhat like a “left-shifted” version of the original.

When run on a waveform that already has minimum phase, such as an IIR filter impulse response, there should be essentially no change in the output from the input--in any of the magnitude, phase, or time domain impulse responses.

Examples of minimum phase usage include “what-if” scenarios to explore the effects of minimizing the delay in waveforms/sequences, reducing FIR filter lengths in real-time systems that are processing-intensive or have high sampling rates, and creating minimum phase versions of impulse responses that have been realized through equalization (transfer function measurement of physical channels).

1.7 Hypersignal-Macro Ex Documentation

1.7.1 Lconv and Lcorr Macro Functions

These macro functions perform long convolution and correlation using an efficient FFT-based frequency domain technique.

After entering ‘lconv’ or ‘lcorr’ inside the Macro Manager menu, the following input paramters are presented: input waveform filename, filter filename, and output waveform filename. lcorr.mac also allows the selection of cross-correlation and circular correlation (enter ‘+’ or ‘+C’, which is identical to entry in the standard Hypersignal time domain function).

Both functions recognize either Hypersignal-generated FIR and IIR filter impulse response files, or arbitrary time domain waveform files. In the case of FIR and IIR filter files, gain is determined by the design of the filter in the FIR or IIR Filter Construction functions. In the case of arbitrary time domain waveforms, unity gain (0 dB) is chosen as the maximum frequency response value of the filter waveform.

Input waveform size is not limited. The current filter length limit is about 200,000 points (based on a maximum internal FFT size of 262,144). Large sizes (i.e. filter size greater than 48,000 pts.) may not run, or not run efficiently, depending on the amount of extended memory in the host machine. See the Protected-Mode Program notes above, especially concerning memory considerations.

Since the protected-mode programs currently can process only real-valued, single-channel input waveforms, you should NOT use lcorr.mac and lconv.mac if the input waveform is complex or two-channel. Use the standard Hypersignal Convolution/Correlation menu selection in these cases. But remember: the standard version has a length limitation of 16,384 samples. Also, even though the standard version can make use of DSP accelerator hardware, it uses a time domain technique, which becomes inefficient as filter lengths increase.

1.7.2 Lfft Macro Function

The lfft macro function performs long FFT size processing on input waveform files. After entering ‘lfft’ in the Macro Manager menu, the following input parameters are presented: input waveform filename, output waveform filename, framesize, and FFT size.

Framesize entry can be used to cause segmentation of the input waveform. When framesize is less than FFT size, remaining FFT input points are zero-filled. A ‘0’ entry causes framesize to be determined automatically from the input waveform file header. This type of operation is identical to the standard Hypersignal FFT Generation function.

FFT size should be entered as a value equal to 2 raised to an integer power, ranging from 4 to 262,144. Non-power-of-two values are quantized to the nearest allowable value. A ‘0’ entry causes FFT size to be determined automatically by the framesize entry (first power of two value greater or equal to the framesize). Large FFT sizes (i.e. greater than 65,536) may not run, or not run efficiently, depending on the amount of extended memory in the host machine. See the Protected-Mode Program notes above, especially concerning memory considerations.

Currently, this function does not allow overlap specification or window selection. If these parameters are needed, use the standard Hypersignal time domain FFT Generation menu selection. Also, this function does not make use of DSP accelerator hardware as does the standard Hypersignal FFT Generation menu item.

1.7.3 Continuous Wavelet Transform

The WLT.MAC function is included with both Hypersignal-Macro Ex and Hypersignal-Acoustic for those who want to experiment with it. This macro is still considered to be in an alpha-state, with improvements continuing to be added.

After entering 'wlt' inside the Macro Manager menu, the following inputparameters are presented: input filename, wavelet specification, output filename, and number of constant scale values.

IMPORTANT NOTE: When entering the wavelet field, wlt.mac accepts one of the following as input parameters:

- A complex wavelet file (enter filename)
- A real wavelet file (enter filename). In this case, a complex wavelet is created by the program, using a Hilbert Transform technique.
- Offset and length of input filename (enter in samples, or points). In this case, the wavelet is taken from signal itself. For example, an entry of '76,32' in this field would cause the program to use a 32-point segment, starting at sample 76, from the input waveform. Note: when using this option, it is suggested that a somewhat periodic segment of the input waveform is chosen.

Some notes about the WLT program:

1) WLT automatically calculates a center frequency of the wavelet using an FFT. Periodic wavelets lend themselves more to this calculation.

2) WLT performs a continuous wavelet transform using a chirp-z transform method developed at the Univ. of Illinois and Rice University. See the wlt.mac file for more details and algorithm/theory references and credits.

3) WLT produces output compatible with Hypersignal frequency domain displays. Currently, the function does not allow the input waveform to be segmented. So as a result, this function does not allow very large input waveforms.

One example wavelet provided on CD is a Gaussian wavelet (gauss.tim) derived from samples 76 thru 106 of a 3-component signal (also provided on CD as wavsig.tim). Note that the last component in wavsig.tim has a center frequency of pi/3 radians (6 samples per period). More example wavelet files provided on CD are daub4.tim, daub12.tim, and daub20.tim, which are standard Daubechies wavelet filters with four (4), twelve (12), and twenty (20) coefficients respectively.

One interesting observation to make using the above example waveforms is a comparison of the wavelet transform and a short time Fourier transform (STFT) computed using a baseband equivalent of the wavelet as a window function. This can be accomplished using the Gauss window option in the FFT Generation function (because the wavelet in this example is Gaussian in shape with a center frequency of pi/3). For example, perform the following steps:

- First run the wlt.mac function using 'wavsig' as input, 'gauss' as the wavelet filename, 'out' as the output filename, and number of scale values as '128'. To do this, select the F3 key, which invokes the Macro Manager menu, and enter 'wlt' on the first line. This will cause a data-entry menu to appear prompting you for necessary entries.
- Second, select FFT Generation in Hypersignal and enter 'wavsig' as the input waveform. Enter the following parameters:

  - FFT order = 8 (FFT size = 256)
  - Framesize = 32 (samples)
  - Overlap = 97 (percent)
  - Window = Ga (Gaussian window)
  -precision and mag/phase approximation fields, if they should apear because you have no DSP hardware accelerator selected, are "don't cares" (they affect accuracy and speed somewhat, and can make a difference when exact results or large waveforms are required)

- Third, enter 'wavsig' and 'out' in the 2-D Spectrograph display to see a comparison. You may want to enable the DUAL AUTOSCALE field in the User Setup area to see better comparison of amplitudes.

Some notes about the above example:

1) STFT processing intended to be compared with wavelet transform output should be done so that the “effective framesize”, or actual number of new points considered in each frame, is 1. This means an overlap value of framesize-1. In the Hypersignal FFT Generation menu, this can be accomplished by entering a percentage value for the OVERLAP field of

$$\frac{\text{Framesize-1}}{\text{framesize}} * 100$$

In the Hypersignal FFT Generation function, one convenient way to verify that the effective framesize is truly 1 is to make sure that the number of frames being processed (displayed at bottom of FFT Gen menu) is the same as the total number of samples in the input waveform (displayed at upper left of main menu).

2) The STFT FFT size should be equal to 2x the number of scale values entered in the wlt Macro Manager menu. Keep in mind that Hypersignal FFT Generation FFT sizes are always a power of 2, because the FFT order is what is actually entered.

3) The STFT framesize should be entered as the length of the wavelet. In the example above, the length of the gauss.tim waveform containing the wavelet was 32. Keep in mind that when framesize is less than FFT size, the Hypersignal FFT Generation function automatically zero-fills the remaining FFT size - framesize points.

4) The window should be chosen to approximate the wavelet being used, which may not be an easy thing to do. In the above example, the wavelet was gaussian, so a baseband equivalent was easy to realize by selecting one of the built-in window types in the Hypersignal FFT Generation function.

5) Enabling DUAL AUTOSCALE may make display comparisons of wavelet transform and STFT results easier. This option causes the waveform with lowest peak value of two displayed waveforms to be scaled up to that of the largest. Currently, we are not sure what to do about wavelet transform output amplitude scaling. For standard frequency domain displays in Hypersignal, a linear magnitude scaling operation is performed such that a 5V pk-pk sine wave in the time domain would show a 5V peak at the frequency of the sine wave in the frequency domain. We have not yet achieved a wavelet transform equivalent scaling operation.

Overall, the choice of what type of analysis method to use (i.e. STFT, wavelet transform, chirp-z, etc.) is highly application- and data-dependent. Many experiments and iterations may be required to determine what works best. If indeed a wavelet transform method is to be used, further experimentation and research is needed to choose an effective wavelet.

1.8 Nihon Version Documentation

The Nihon versions of Hypersignal-Macro, Hypersignal-Acoustic, and Hypersignal-Macro NRT are essentially the same as Hypersignal-Macro, except portions of the internal interfaces to disk, video, and keyboard have been modifed to run on PC-980x machines manufactured by NEC and used commonly in Japan. Differences in keyboard, video, and on-line Help usage are pointed out where necessary throughout the documentation.

this page intentionally left blank

2.0 GETTING STARTED

NOTE: **It is *highly recommended that you read this section and at least skim chapter 3, USING HYPERSIGNAL, before actually running any Hypersignal software. If you absolutely must begin experimenting with Hypersignal immediately, follow the instructions below.***

2.0.1 Quick-Start

This section is reluctantly provided for those users who simply refuse to do what's good for them and read chapters 2 and 3 before using the software. The instructions below will get you up and running in Hypersignal; but after that, you will have to refer to the on-line Help (click on "User Guide" icon in Signalogic DSP Software folder or group, or use the 'F1' key inside Hypersignal software), this manual (Users Manual), and the Software Reference Guide when you have questions.

2.0.1.1 Quick-Start from CD

To install Hypersignal software from CD, do the following:

1) Insert Signalogic software CD into CD-ROM drive D:\SETUP

Win 9x: From Start Menu select "Run", and enter D:\SETUP where D

denotes your CD_ROM. Click OK.

Win 3.1x: From Windows program manager select File|Run, and enter D:\SETUP as appropriate. Click Ok.

2) Follow instructions shown by setup software.

After starting the setup software, you should see a dialog box indicating which software items will be installed. In this dialog box, you should see at one of Hypersignal-Macro, Hypersignal-Macro Ex, or Hypersignal-Acoustic software selected, and possibly additional DirectDSP, Real-Time Composer, Visual Bench or DSPower software items selected as well.

2.0.1.2 Quick-Start from Floppy Disk

1) Copy *.* from distribution diskettes #1 and #2 to a subdirectory named "HSMACRO" (or similar)

2)Copy *.* from distribution diskette #3 and #4 to a subdirectory named "WAVEFORM" under "HSMACRO"

3) Change to the "HSMACRO" subdirectory and type in one of the following batch files (which one you have depends on your software version):

HS <ENTER>	(Hypersignal-Macro Non-Real-Time)
HSM <ENTER>	(Hypersignal-Macro)
HSA <ENTER>	(Hypersignal-Acoustic)
HSN <ENTER>	(Hypersignal-Macro NRT, Nihon Version)
HSMN <ENTER>	(Hypersignal-Macro, Nihon Version)
HSAN <ENTER>	(Hypersignal-Acoustic, Nihon Version)

4) select a graphics driver when prompted—use the "probe best-guess" if you are unsure of what video hardware is in your system

5) select a mouse driver when prompted—use the "probe best-guess" if you are unsure of what mouse hardware is in your system

6) set the subdirectory field in the User Setup menu selection to 'WAVEFORM' (the User Setup menu function is under the Utilities column, at the righthand side of the main menu)

7) start using Hypersignal functions—answer filename prompts with the example waveform filenames included on distribution diskette #3—some example filenames are: upchirp, speech, elliptic, and signal

2.0.2 What Documentation and Software You Should Have

Make sure you received the following items in your shipment:

1) Hypersignal-Macro Users Manual (this manual)

2) Hypersignal-Macro Software Reference Guide

A separate manual that provides a brief, concise list of all data-entry menu fields and interactive display commands for all Hypersignal functions.

3) Hypersignal-Macro Tutorial

A separate manual that includes several practical, step-by-step examples that encompass most of the functions in the software.

4) Hypersignal-Macro Hardware Reference Guide

Provided with real-time versions of the software only, this separate manual contains specific information on every DSP/Acquisition board that is directly supported. This manual contains detailed individual sections organized by board manufacturer and type of board.

5) CD or (Floppy Disk)

"Signalogic DSP Software" CD, with one or more boxes marked "Hypersignal-Macro", "Hypersignal-Macro Ex", or "Hyper -sinal-Acoustic" checked.

In case of legacy system, four 1.44 Mbyte floppy diskettes, referred to as "distribution diskettes". Distribution diskettes #1 and #2 contain the basic Hypersignal and DSP/Acquisition (real-time versions only) software, various utilities and support files, example user-defined external programs, and several ready-to-execute macro functions. Distribution diskette #3 and #4 contains example waveforms. If you have received an update version of the software, then you may not have received distribution diskette #3 or #4, based on the assumption that you already have several example waveforms.

NOTE: **If any of the above items are missing, contact your distributor immediately.**

2.0.3 Registering as a User to Ensure Technical Support

Near the front of this manual (on pages v and vi) are USER REGISTRATION and USER FEEDBACK forms, each on one side of the same physical page. We strongly encourage you to fill out the user registration form and send it back. With the information you provide, we can enter these in our database as part of the original purchase order record. Then, should you need any technical support from us over the phone, it is much easier for us to figure out exactly what items and configuration you have (especially if you are not the original customer), and give you the best support possible. We highly value your comments, suggestions, and criticism of the software, so please try to take the time to fill out and return the user feedback form also (other side of registration form).

2.0.4 Not Making Unauthorized Copies of Your Software

You can make as many backup copies of the software as you wish for your own use. However, use of a "single-user" copy of the software by more than one person at the same time implies "unauthorized" copies of the software have been made *and is illegal* (see license agreement at beginning of this manual). Whether your software is a "single-user" or "site-license" (multi-user) version is indicated by the "personalized display area", located in the center of the Hypersignal main menu.

If you desire an authorized change to the personalized display area on the main menu due to address change, spelling error, etc., contact your distributor to obtain replacement diskettes incorporating the required changes.

NOTE: **If you attempt an unauthorized change to the personalized display area, the Hypersignal program will detect it, provide an audible and visual message, and stop in a normal, non-destructive fashion. For anyone with intentions to defeat the personalized display area with a hardware emulator, be prepared to spend time: there are multiple, random encryption calculations *and* comparisons that must be removed, and relevant data is neither contiguous nor viewable in a sensical form. The code locations of that percentage of calculations and checks that are active at any given time change every time the program is run.**

2.1 Installing Hypersignal

NOTE: **The instructions below assume that you are installing the software onto a hard disk. The standard versions of Hypersignal occupy too much disk space to be used in a 1.2 Mbyte floppy-disk-only configuration. There is an "academic version" of Hypersignal-Macro NRT (Non-Real-Time) software available that can be used in a floppy-disk-only system. If you are using this version in a floppy-only system, insert distribution diskette #1 into a 1.2 Mbyte floppy drive, log to that drive, and skip to the next section, Running Hypersignal.**

Below is the basic step-by-step sequence for installing Hypersignal software on your hard disk:

1) Copy ∗.∗ from distribution diskettes #1 and #2 to a subdirectory named " HSMACRO HSIGNAL" (or similar)

2) Copy ∗.∗ from distribution diskette #3 and #4 to a subdirectory named "WAVEFORM"

2.2 Running Hypersignal

From Windows: Clicks Start, Program, Signalogic DSPsoftware, Hypersignal Macro icon.

Although you can directly run Hypersignal by invoking the appropriate "HSxxxxx.EXE" file, we suggest that you use the batch file we have provided. The batch file first installs the "Metagraphics graphics driver" (makes it memory-resident), runs Hypersignal, and then removes the graphics driver from memory when you exit Hypersignal. This will keep the graphics driver out of your system memory when you are not using Hypersignal. Below is the basic sequence for running Hypersignal:

1) Change to the "HSMACRO HSIGNAL" subdirectory (or whatever name you used when installing the software—see previous section 2.1, Installing the Software) and run one of the following batch files (which one you have depends on your software version):

HS <ENTER>	(Hypersignal-Macro Non-Real-Time)
HSM <ENTER>	(Hypersignal-Macro)
HSA <ENTER>	(Hypersignal-Acoustic)
HSN <ENTER>	(Hypersignal-Macro NRT, Nihon Version)
HSMN <ENTER>	(Hypersignal-Macro, Nihon Version)
HSAN <ENTER>	(Hypersignal-Acoustic, Nihon Version)

NOTE: **When you run Hypersignal for the first time, you will see a message of the form "HSxxxxx.CNF file not found — initializing new configuration...". All this means is that you must set up driver selections and other basic information for the first time before the software is fully operational. When you "exit" Hypersignal, all selections and configuration will then be saved. When you next run the software, execution will skip the setup selections.**

2) Select a graphics driver (see section 2.4.1, Selecting a Graphics Driver, below for full discussion). Generally, using the "best-guess probe" value provided by the software will produce reasonable results—better resolution may be possible if you know exactly what kind of video board is in your system.

3) Select a mouse driver (see section 2.4.2, Selecting a Mouse Driver, below for full discussion). Using the "best-guess probe" value provided by the software may or may not produce reasonable results. The probe accurately detects the presence of a memory-resident MOUSE.COM driver and some mice connected directly to the COM1 port, but has trouble with other mice and

with COM2 port connections. Best results are almost always obtained if you have previously installed the MOUSE.COM driver in your system memory, and then enter "5" for the memory-resident driver selection. This method allows the mouse manufacturer's own driver, which should be the best driver in existence for that mouse, to do most of the work. If a MOUSE.COM driver has not been installed, then good results are also usually obtained with Microsoft or Logitech mice connected directly to the COM1 port. In this case, enter '4C1' or '2C1', respectively.

If you have no mouse hardware in your system, then enter '7' for the keyboard-only selection. If you don't make this entry, then the software will continue to display an initial message of the form "no mouse hardware found..." every time it executes.

NOTE: The following steps assume that you know how to make basic menu selections and enter data into menu fields. If not, please take a quick look at section 3.1, Keyboard and Mouse Interface.

4) Set the subdirectory field in the User Setup menu selection to WAVEFORM (or whatever name you used for distribution diskette #3 files when installing the software).

5) If you have a real-time version of software, select the System Configuration menu item and enter the designator for your DSP/Acquisition hardware (see section 2.5.2 below, Initializing the System Configuration Menu, for full discussion) in the first four fields. The on-line Help for this menu item contains a full list of hardware designators. Selecting DSP/Acquisition hardware is only necessary to run those Hypersignal functions requiring external analog input. Functions that can use the DSP as an accelerator will still operate using host processor until a DSP board has been selected.

6) For both real-time and non-real-time software, enter the DEVELOPMENT DSP field in the System Configuration menu selection (last field, at the bottom of the menu). Valid entries for the DEVELOPMENT DSP field include:

'AD21'	Analog Devices ADSP-210x and ADSP-2106X AD6D
'WE32' 'WE32C'	AT&T DSP32 and DSP32C
'M56' 'M96'	Motorola DSP56xxx and DSP9600x
'N20'	NEC 77220
'TI10' 'TI20' 'TI25' 'TI30' 'TI40' 'TI54'	Texas Instruments TM320C1x, TMS32020, C25, and C3x, C4x and C54xx

This field controls subsequent code generation and file acquisition from DSP manufacturer development tool output files. Entering this field is not required for the software to run—in fact, leaving it as a 'NONE' entry may be useful for DSP chip-independent experimentation as you design your DSP algorithms.

7) You're up! You can now run Hypersignal functions by entering filename fields with the example waveform filenames included on the distribution diskettes—some example filenames are upchirp, speech, elliptic, and signal.

2.3 Using a Math Coprocessor

For Pentium, Celeron, and Pentium I, II, III systems, is assumed as part of the Pentium or MMX processor.

For legacy (e.g. 486), Hypersignal software performs automatic detection and utilization of the 80x87 math coprocessor. If a math coprocessor is present in your legacy system, then the software should detect exactly which one (8087, 80287, 80387 or 80487), and set up for use of the math coprocessor where appropriate in Hypersignal. If a math coprocessor is not present in your legacy system, then all functions that require floating-point operations (such as Filter Construction, Difference Equations, and many others) are still performed with the same precision, but much slower (about 4 to 6 times slower). The decrease in speed occurs because software routines are used that exactly emulate the 80x87 math coprocessor. These software routines are much slower, because many 80x86 instructions are required to do what the 80x87 normally does in one.

If, because your legacy system's 80x87 chip is faulty or for some other reason, you wish to override the automatic 80x87 detection then type at the DOS command line (before executing Hypersignal):

SET 87=N <ENTER> (sets the DOS environment variable "87" to OFF; forces the software to use 80x87 software emulation)

SET 87=Y <ENTER> (sets the DOS environment variable "87" to ON; forces the software to use 80x87 hardware)

If you wish to re-activate automatic detection of the math coprocessor by the Hypersignal software, then type at the DOS command line (before executing Hypersignal):

SET 87= <ENTER> (undefines the environment variable; returns software to automatic detection)

Most Hypersignal operations that require a high-degree of numerical precision use 8-byte precision. However, arbitrary filter design and some Difference Equation functions use 10-byte precision (also emulated exactly through software if an 80x87 is not present).

2.4 Selecting Graphics and Mouse Drivers

Upon executing Hypersignal for the first time, or selecting Graphics/Mouse Config from the main menu, two separate lists will be presented to you: first, a list of graphics drivers, and then a list of mouse drivers. If you have selected Graphics/Mouse Config from the main menu, then you can use the 'Esc' key to exit at any time, but if you are executing the software for the first time, you *must* choose graphics and mouse drivers before the software will allow you do anything else.

When you next "exit" the software ('E' key from the main menu), the current driver selections are saved in the Hypersignal environment configuration save file, and no further graphics or mouse driver selection is required unless you choose the Graphics/Mouse Config selection from the main menu or receive a new version of software.

Hypersignal uses the Metagraphics "MetaWindow" memory-resident driver to provide direct support for over 100 different types of graphics hardware and 6 different types of mouse hardware. Supported graphics hardware includes CGA, EGA, VGA, PS/2, Hercules, AT&T, various portables, Wyse, and several popular 800x600 and 1024x768 formats. Supported mouse hardware includes Microsoft, Logitech, Mouse Systems, and others. Also supported directly is the NEC PC-980x series of machines and compatibles, which are widely used in Japan (the Nihon versions of Hypersignal-Macro, Hypersignal-Macro NRT, and Hypersignal-Acoustic are designed specifically to run on these machines). MetaWindow is manufactured by the Metagraphics Software Corporation—see the copyright notice at the beginning of this manual.

2.4.1 Selecting a Graphics Driver

NOTE: Don't forget the 'PgUp' and 'PgDn' keys! Over 100 different types of graphics driver selections are available. These are displayed in alphabetical order on 4 (four) different screen pages. If you don't see a certain driver selection at first, use the PgUp and PgDn keys to view all possible selections.

The software always performs a "probe" into your system hardware and returns a "best-guess" suggestion as to what graphics driver choice should work. This probe is intended to serve only as a guide, and need not be followed in all cases. For example, if you had an 800x600 or 1024x768 resolution "SuperVGA" board in your system, the probe might return selection 42, which indicates standard 640x480 VGA hardware. Of course, this selection would work fine, but you could get better resolution by choosing the appropriate 800x600 or 1024x768 selection, depending on the manufacturer and model of the particular video board in your system.

NOTE: If you specify a color graphics driver (CGA, EGA, VGA, etc.) to be used with a *monochromatic* monitor, then add an "M" suffix to your selection; e.g. enter '42M' or '42m' for a VGA-compatible video card connected to a black and white monitor. Some monochromatic monitors, such as Compaq, will have excellent response to color displays, while others will not. The "M" suffix affects only menus, on-line Help, and other text displays, reducing the number of "color", or gray-level, values used in these displays.

If you have selected Graphics/Mouse Config from the main menu and end up making no change in the current graphics driver selection (a blank entry will use the current selection, which is displayed along with the "probe best-guess" at the top of the display), then all waveform display settings will remain unchanged. However, after running Hypersignal for the first time, or after specifying a different graphics driver selection, all waveform display settings will be initialized to a default state. Altering only the monochromatic "M" parameter will not affect current waveform display settings.

2.4.1.1 Graphics Driver Trouble-Shooting

If no graphics hardware is present in your machine, a message will be displayed indicating that no hardware whatever could be found, and you will be unable to specify a driver selection. Basically, this condition will deactivate all waveform and instrumentation displays in the Hypersignal software. If you feel that this message is being displayed for your system incorrectly, then contact your software distributor immediately—there may be some sort of hardware or BIOS conflict that can be resolved.

If your graphics driver selection is not producing desired results, or you are unsure of just what type of video hardware is in your system, then the following discussion may help. As you may be aware, many types of graphics hardware will support more than one type of driver listed. Also, some hardware may be available with varying amounts of memory. If displays appear to use only half the screen, or the display "wraps-around", try a driver selection for the same type of hardware, but with less vertical resolution. You may have to experiment to determine the choice that works best. If you are still unsure, try the following guesses—they represent four of the most commonly used types of hardware:

PC/XT compatibles	34 (CGA color), 32 (Hercules mono)
AT compatibles other	286 machines 37(EGA color)
386 and 486 machine	42 (VGA color)

PS/2 compatibles 48 (VGA color), 46 (MCGA mono)

Some driver selections listed are for video hardware that use the "secondary display adaptor" address as their default base address. Generally, such video hardware has some distinguishing characteristic about it that implies that it may be used in a system as the second video display, in addition to a standard CGA, EGA, or VGA color display. Extremely high resolution (i.e. greater than 1200x1200) and monochromatic (black & white) operation are two common types of such hardware. When one of these boards is used in a system, it is easy to choose a driver selection that causes the wrong display base address to be used. Symptoms of a driver choice that assumes an incorrect base address include displays not being restored after striking the 'Esc' key, and unwanted text or strange "graphics" characters appearing in waveform display modes. If you are unable to get the software working correctly with non-standard graphics hardware that is listed as a driver selection, please contact your distributor, as problems of this nature can usually be resolved.

Some "compatible" machines do not always react to all combinations of text, graphics, and color as they should. For example, plasma or some LCD display machines must convert all color combinations to on, off, and possibly an intermediate shade. Usually they do this by maximizing contrast—black on white and white on black are translated equally well. However, sometimes the scheme used by the manufacturers does not always work flawlessly, especially in the case of colors that are not necessarily "dark" or "light". Try entering the 'M' monochromatic suffix if problems of this type should occur.

NOTE: **Graphics driver selections for video hardware with 640x480 resolution or less typically require the use of one of the graphics font files provided on the Hypersignal distribution diskettes. These files have ".FNT" filename extensions. If no text is displayed in waveform displays or you think the font used is too small or too large, make sure that all .FNT files present on the original distribution floppies are in your working Hypersignal subdirectory. Graphics hardware with resolution greater than 640x480 generally use "bitmapped vector-generation" text, which requires no graphics font files.**

NOTE: **If you are using an AT&T DEB video board, the AT&T DEB driver file, "DEDRIVER.DEV", must be included in the system CONFIG.SYS file.**

2.4.2 Selecting a Mouse Driver

The software always performs a "probe" into your system hardware and returns a "best-guess" suggestion as to what mouse driver choice should work. Even more so than the graphics driver, this probe is intended to serve only as a guide, and need not be followed in all cases. For example, the probe accurately detects the presence of a memory-resident MOUSE.COM driver and some mice connected directly to the COM1 port, but has trouble with other mice and with COM2 port connections. Best results are almost always obtained if you have previously installed a MOUSE.COM driver (provided by the mouse manufacturer) in your system memory, and then enter '5' for the memory-resident driver selection. This method allows the mouse manufacturer's own driver, which should be the best driver in existence for that mouse, to do most of the work. If a MOUSE.COM driver has not been installed, then good results are also usually obtained with Microsoft or Logitech mice connected directly to the COM1 port. In this case, enter '4C1' or '2C1', respectively.

If you have no mouse hardware in your system, then enter '7' for the keyboard-only selection. If you don't make this entry, then the software will continue to display an initial message of the form "no mouse hardware found..." every time it executes.

NOTE: **If the "Keyboard Only" selection is made in the Hypersignal software, then the "repeat rate" that controls the rate at which characters are displayed when a key is held down on the keyboard, sometimes called "typematic action" in IBM documentation, is temporarily increased to its maximum during Hypersignal operation. This makes the zoom windows and filter design cursors much more usable when using only the keyboard, especially for slower machines (8-16 MHz AT clones).**

2.4.2.1 Mouse Driver Trouble-Shooting

After installing the MOUSE.COM driver, you will probably not be able to go back to operating the mouse directly through the COM1 or COM2 ports without first turning your machine all the way off (not just resetting or rebooting). This is a known problem with the Metagraphics driver, and has to do with the way the COM1 and COM2 serial ports are initialized.

When you move the mouse very rapidly, if menu selection or target window display seems to "wrap around" or actually go in the opposite direction, you may be able to mitigate this effect by adjusting "speed" parameters in your MOUSE.COM driver. Most MOUSE.COM drivers allow you to control the maximum rate of interrupts to application software using parameters on the MOUSE.COM command line; lowering the rate of interrupts may help with any rapid mouse movement problem.

2.4.3 Installing the Metagraphics Driver Above Conventional Memory

Normally, the Metagraphics graphics driver is invoked by the "HSx" batch file, which makes the driver memory-resident before running Hypersignal, and then removes the driver from memory afterwards. However, the Metagraphics driver can also be treated as a DOS device driver, which allows the terminate-and-stay-resident (TSR) portion of the driver to be loaded via the system "CONFIG.SYS" file and then subsequently relocated above conventional memory (above 640 kByte but below 1024 kByte). The relocation should be done using the LOADHI command provided by various expanded memory managers, such as QEMM-386. Relocating the Metagraphics driver above conventional memory will free up approximately 113 kByte of additional memory for the use of "user-defined" external programs (see section 3.6, Invoking External Programs and the DOS "Hot-Key") during Hypersignal operation. If memory above 640 kByte is not available in (1) contiguous 113 kByte block, other device drivers and DOS resources, such as file and buffer configuration commands, may be relocated to free up conventional memory for user-defined external program operation.

NOTE: **All DSP and adapter boards should be installed in the computer before running the installation program provided with the various expanded memory managers, which should then detect the presence of installed boards and attempt to avoid memory conflicts with this hardware during setup.**

2.5 Basic Software Setup

This section assumes that the software is already up and running, and that you know how to choose menu items and enter data in menu fields using the data-entry editor. If not, please skim section 3.3, The Hypersignal Main Menu, and section 3.4, Using the Data-Entry Editor, before continuing.

Basic software setup consists of making some initial selections in the Utilities main menu column.

2.5.1 Initializing the User Setup Area

Select the User Setup menu item. The drive field should already be set for you. The subdirectory field will be set to whatever was current when you first ran Hypersignal—you will want to change this to be your "WAVEFORM" subdirectory. The TIME FILENAME and FREQ FILENAME fields don't really matter, since you can enter these filenames in all functions anyway. If you don't like programs beeping and buzzing at you when you do something they don't like, then set the SOUND to 'OFF'. Everything else should be left at the default. Pressing the 'Esc' key or pressing the 'Enter' key (or pushing any mouse button) while the cursor is on the last field will leave this function.

2.5.2 Initializing the System Configuration Menu

(Selecting a DSP/Acquisition Board Driver)

Select the System Configuration menu item. If you have Hypersignal-Macro non-real-time software, then only the very last field in this menu, DEVELOPMENT DSP, is active—all other fields are ignored. For this reason, the DEVELOPMENT DSP field is covered first in the menu field descriptions below.

DEVELOPMENT DSP:

For all versions of software, set this field according to the DSP chip you will be generating code for and/or acquiring development tool files for (development tool files are files created by the chip manufacturer's development utilities, such as software simulator, or hardware emulator).For a list of valid entries in the DEVELOPMENT DSP field. Please refer to Section 2.2 **Running Hypersignal**.

ANALOG CONVERSION:

For real-time software versions only, this field selects the DSP/Acquisition hardware that will be used for A/D and D/A analog conversion performed by the Analog Conversion menu selection. See the on-line Help for a complete list of possible "hardware designators" that can be entered in this field. Default entry is 'NONE', which will cause access to the Analog Conversion function from the main menu to be denied.

For hardware with only A/D capability, D/A output will default to the DSP ACCELERATOR field entry (see below).

See the Hypersignal-Macro for a detailed discussion of individual boards, as well as hardware considerations such as method and maximum rate of data transfer, anti-aliasing, multichannel operation, and programmable hardware features.

DSP ACCELERATOR:

For real-time software versions only, this field selects the DSP hardware that will be used to accelerate DSP functions. See the on-line Help for a complete list of possible "hardware designators" that can be entered in this field.

Note that a few of the hardware designators listed in the on-line Help may not be valid, if they refer to "pure acquisition" hardware that has no onboard DSP device. Default entry is 'NONE', which will cause the host processor to be used instead.

During normal processing, a 'NONE' entry is perfectly valid, essentially deactivating the DSP accelerator, reducing processing to host processor speed levels, and making available the standard precision choices. In fact, when using fixed-point DSP hardware, the 'NONE' selection may be desired for some types of FFT operations, such as filter impulse responses, where time required to process the input waveform is not a factor and high precision is needed.

DIGITAL OSCILLOSCOPE:

For real-time software versions only, this field selects the DSP/Acquisition hardware that will be used for the Digital Oscilloscope function. See the on-line Help for a complete list of possible "hardware designators" that can be entered in this field. Default entry is 'NONE', which will allow only waveform files as input (no analog input).

The DSP/Acquisition hardware chosen must have multichannel capability in order to see a true dual-channel display, although different *dual-trace* display formats are possible from one analog input channel. See the Hardware Reference Guide for a detailed discussion of individual boards, as well as general discussion of various hardware considerations, such as sampling rates, triggering, real-time filtering, and DSP source code modification.

SPECTRUM ANALYZER:

For real-time software versions only, this field selects the DSP/Acquisition hardware that will be used for the Spectrum Analyzer function. See the on-line Help for a complete list of possible "hardware designators" that can be entered in this field. Default entry is 'NONE', which will allow only waveform files as input (no analog input).

The DSP/Acquisition hardware chosen must have multichannel capability in order to see a true dual-channel display, although different *dual-trace* display formats are possible from one analog input channel. Intrachannel display options include simultaneous magnitude, phase, time, power spectra, and filtered vs. unfiltered. See the Hardware Reference Guide for a detailed discussion of individual boards, as well as general discussion of various hardware considerations, such as sampling rates, triggering, real-time filtering, and DSP source code modification.

DSP Clock and External Sampling Clock Entry

After the above four fields, an '-NN' DSP clock rate entry is possible. NN should be a value in MHz, and is intended for DSP/Acquisition boards that:

1) base their analog sampling rate on the main DSP clock, typically using an integer division, or submultiple method

2) are available in more than one clock rate version

For example, the some TMS320C25-based in more than one speed, with the DSP typically running at 28 MHz, 40 MHz, or 50 MHz clock rates. Thus, you might enter 'XXXXX-40', where "XXXXX" is the hardware designator, to specify the 40 MHz version. As another example, some TMS320C30 boards come in 33 and 40 MHz versions. The software will assume a default rate indicated in the "Performance Specifications" section of the hardware's entry in the Hardware Reference Guide if no '-NN' entry is specified. In most cases, the default is correct, since there are still relatively few boards that come in different speeds. For this reason, no entry (default) is highly recommended unless you are absolutely sure of the DSP clock rate.

Also, an '-EXT' external clock "attribute" can be entered after any hardware designator in the above four fields, *where in this case "clock" is referring to the analog sampling rate clock, and not the DSP clock.* External clock entry deactivates any assumed sampling rate programming for the entered hardware, allows menu entry anywhere from 0 (dc) to 1 GHz in the Analog Conversion, Digital Oscilloscope, Spectrum Analyzer, and D/A playback-from-display functions, and passes a 16-bit or 32-bit integer value equal to the menu entry directly to the DSP source code. Do not attach this attribute for boards that do not truly have some sort of external clock input—otherwise they will "freeze up". Use this entry only if you are sure that the hardware's analog sampling circuitry is operating from an external clock. External clock entry can be made additionally to '-NN' entry described above.

NOTES: **If you have a real-time version of Hypersignal, then you should enter in the first four fields the designator for the current DSP/acquisition board you have in your system. See the on-line Help for a list of designators for different manufacturers and board types.**

There are only two cases where you would not have the first four fields all the same: you have two or more supported boards in the system, or you purposefully want to disable the DSP ACCELERATOR, in which case a selection of 'none' will cause the host processor to be used instead. A situation for the latter would be for analog hardware that has no DSP chip, in which case either 'none' or some other DSP board selection would have to be entered for DSP ACCELERATOR. If you are using an analog-only board with no DSP chip, or no D/A playback capability, then whatever board you enter in the DSP ACCELERATOR field will assume those responsibilities. For example, if you are using an analog-only board for your SPECTRUM ANALYZER selection, then you must have a valid DSP ACCELERATOR selection working in order to run the spectrum analyzer, since the real-time Fourier analysis can't be done on your analog input board.

MEMORY BASE ADDRESS and I/O BASE ADDRESS:

These two fields, MEMORY BASE ADDRESS and I/O BASE ADDRESS, are somewhat more complicated. Normally, you will want to leave these two fields at their default (default values are listed in the on-Help and in the Hypersignal-Macro). However, there can be times when you need to change base addresses—the best examples being a possible conflict with boards you already have in your system, or a manufacturer that ships different boards with different base addresses.

Note that there is one entry for most manufacturers supported—not for every board. If you need to change a base address, *be sure you only change the value for the relevant manufacturer*—the other values do not need to be correct, but *they must remain entered!* Enter zeros for the other manufacturers if you must—but enter one value for every manufacturer listed in the field label (you can always get the defaults back by deleting the configuration file, if necessary).

If the manufacturer of your DSP/Acquisition hardware does not appear on the menu, then you will have to change memory and I/O base addresses directly in the user-environment configuration file. See section 3.7, Accessing and Updating the User-Environment Configuration File, and also the "CONFIG.DOC" file, included on distribution diskette #2, for instructions on how to do this.

Multiple Boards From Same Manufacturer

If you have two or more boards in your system from the same manufacturer, then Hypersignal will consider the first board mentioned in the four "operation" fields (ANALOG CONVERSION, DSP ACCELERATOR, DIGITAL OSCILLOSCOPE, and SPECTRUM ANALYZER) to be at the entered I/O base address, the second board to appear in the operation fields to be at the I/O base address plus 10H (16), the third board at the I/O base address plus 20H, and so on. Note that the software has no way of knowing that two boards in the system at the same time unless you have entered both designators at least once in the first four fields of the System Configuration menu. If only one is entered then the software assumes that this board is located at the corresponding base address value, regardless of how you have set the jumpers on the board.

NOTES: **All memory and I/O base addresses must be entered in hexadecimal. Please consult the Hardware Reference Guide to see if your DSP/Acquisition hardware requires a base address entry other than the default before it will operate. Remember, some manufacturers send out different boards with different base addresses.**

All memory and I/O addresses *must not conflict when different boards are present in the system at the same time. If you change the base address jumper(s) on a board from the factory configuration, you must also change the appropriate memory or I/O base address field.*

The "Exit,Save" function ('E' key) from the main menu will save all entries for use in later sessions.

A few supported hardware manufacturers use a separate, memory-resident driver that requires any base address setup to be done outside of Hypersignal (usually via a driver program command-line switch or a DOS environment variable). Consult the Hardware Reference Guide section for your hardware to see if this is the case.

80286/80386 INSTRUCTION SET:

This field should generally be set to 'Y' if you have an 80386, or better (e.g. Pentium) machine. Depending on the DSP/Acquisition hardware design and data transfer scheme implemented by the hardware manufacturer, this field may or may not be relevant. Only change it to 'N' if you definitely have an 8086 or 8088 (IBM PC/XT machine) or you are specifically instructed to do so in the Hardware Reference Guide, depending on your particular DSP/Acquisition hardware.

Symptoms of leaving this field set to a 'Y' when not appropriate for certain DSP/Acquisition hardware include "spikes" in your analog input data, or board freeze-ups during real-time operation.

DEVELOPMENT DSP:

The last field, DEVELOPMENT DSP, was covered first (above), since it is the only valid field in the System Configuration menu for Hypersignal-Macro non-real-time users.

Entering System Configuration Fields

Pressing the 'ESC' key, pressing <CTRL-ENTER>, or pressing the 'ENTER' key(or any mouse button) while the cursor is on the last field will leave this function, and enter all current selections.

2.5.3 Initializing the External Program Configuration Menu

Finally, you might like to set the software up for one or two external programs. For example, you might want to set up a text-editor as a menu item under the Utilities column, so that you can go directly to your editor from Hypersignal to review generated source code, or set up a debugger so that you can see what Hypersignal has downloaded and is executing on your DSP board. To do this, select the Ext Program Config menu item. In the appropriate field for where you want the external program menu label to appear on the Hypersignal main menu, enter the filename of your program, followed by a comma, followed by the menu label as you would like it to appear. For instance, an editor might be set up by entering

editor,text-editor

in the UTIL FILE, MENU LABEL 1 field. Such an entry would cause Hypersignal to look for an executable file of name "editor.com" first, then "editor.exe", then "editor.bat", in that order, whenever the menu item "TEXT-EDITOR" is selected from the Hypersignal main menu. Of course, you can explicitly enter an extension if not ".COM", ".EXE", or ".BAT". Be sure to specify where the external file is located in the DRIVE and SUBDIRECTORY fields, although Hypersignal will look elsewhere, including the DOS path list, for the file if not found immediately.

NOTE: **Hypersignal updates the user-environment configuration file before executing any external program (including a DOS shell invoked by the 'F10' key) and reads the configuration file back again after the external program finishes executing. In this way, a user-created external program can access and modify the configuration file, causing changes in the Hypersignal environment upon return. In order to guarantee that the configuration file can be read after the external program (or DOS shell) returns, Hypersignal saves the current DOS path (current logged drive and/or subdirectory) before yielding control, and restores the saved value afterwards. If you wish to override the automatic restoration, because external or macro processing needs to change DOS paths, you can set the RESTORE DOS PATH field in this menu to 'N'. However, keep in mind that if you disable automatic restoration, and then change the current logged drive and/or subdirectory while in an external program or DOS shell, Hypersignal will read (or attempt to read) a new configuration file, if any, after the external program finishes.**

Pressing the 'ESC' key or pressing the 'ENTER' key (or pushing any mouse button) while the cursor is on the last field will leave this function.

See section 3.6, Invoking External Programs and the DOS "Hot-Key", for more detailed information about user-defined external programs.

2.6 32-Bit Function Notes

Hypersignal processing functions that use 32-bit code or large amounts of memory have been implemented as protected-mode programs controlled by macro functions. Examples include the tim2wav conversion utility (distributed with HS-Macro, HS-Macro Ex, and HS-Acoustic) long FFT, convolution, and correlation (distributed with HS-Macro Ex and HS-Acoustic only) MLS stimulus & response (distributed with HS-Acoustic only), and several others. You should have at least 8 Mbyte of extended memory in your system if you want to use these functions. The DOS extender program dos4gw.exe is used to load and run the protected-mode programs. Once running, the protected-mode programs request extended memory based on processing needs. Some important notes about amount of extended memory and its usage:

1) The amount of memory installed in your machine is not the amount available. The first 1 Mbyte is devoted to "conventional" memory and housekeeping machine requirements. Also, other programs use extended memory, such as smartdrv.exe, which is commonly installed with later versions of DOS and Windows, and which takes up 2 Mbyte of extended memory in its default mode. Finally, expanded memory may take a chunk, too—although expanded memory is used less often with state-of-the-art systems. To determine if this is the case, look for the line "device=EMM386.EXE ..." in your system's config.sys file. Thus, a machine with only 4 Mbyte of extended memory often ends up with not much available for user programs. You can run the pminfo.exe program included with Hypersignal to determine the amount of extended memory actually available.

2) Several Hypersignal functions use "protected-mode" programs during processing. These programs need the dos4gw "DOS Extender" in order to run correctly. Dos4gw requires a piece of conventional memory (DOS memory less than 640k) in order to run get started; after which, it runs the protected-mode program in upper memory (above 1Mb). Depending on what drivers and resident programs are loaded in your machine (i.e. utilities specified in your config.sys and autoexec.bat files), there may not be enough conventional memory to run protected-mode programs fully. You may see a message such as "not enough memory" or "external program problem". In such cases, make sure that as many drivers and resident programs are loaded into upper memory as possible. Specifically, make sure that your config.sys file contains the follwing lines near the beginning of the file:

```
Device=C:\dos\himem.sys
DOS=HIGH,UMB
device=C:\dos\emm386.exe NOEMS ....
```

Note that the emm386.exe line may have other items after the NOEMS. The above lines will ensure that utilities are loaded into high memory. You should have about 550k free to run Hypersignal and protected-mode programs successfully. You can check this by using the "mem" command from the DOS prompt.

3) The protected-mode programs can be run under Windows (i.e. if Hypersignal has been launched directly from Windows, or run from a DOS command prompt that was launched from Windows). When run under Windows, the protected-mode program's use of system memory is fully compatible with Windows and whatever Windows applications are already running, including access to Windows virtual memory, which extends the amount of memory available on a system by swapping unused sections to disk file. *However, do not expect virtual memory to help with typical dsp programs.* For example, lcorr.exe, lconv.exe, and lfft.exe, all are nearly worthless when they use virtual memory, because the bit-reversed memory-access nature of the FFT routine used by these programs causes excessive "disk thrashing". Disk thrashing occurs when vitual memory is being swapped in and out from disk file so frequently that programs that should take seconds end up taking hours. To run larger FFT sizes, larger filter sizes, and larger MLS stimulus & response sizes, it is strongly suggested that you upgrade (expand) the amount of memory in your computer. For example, on a typical 486 system running Windows 3.1, and with smartdrv.exe installed in the autoexec.bat file and no EMS (expanded) memory installed, 8 Mbyte of extended memory typically would allow a 262,144 pt. complex FFT in the lfft.mac function without disk thrashing. But using the lcorr.mac or lconv.mac functions with an implied FFT size of 262,144 (i.e. a filter size of 200,000 pts.) would require a full 16 Mbyte, due to the extra memory needed for convolution and correlation.

A note about DOS versions less than 5.0: dos4gw.exe still needs some conventional memory to get started--before loading a protected-mode program--and this can be a problem with DOS versions less than 5.0. In these cases, the combination of Hypersignal, the met_shel.exe graphics driver, DOS's command.com, and other utilities can occupy enough conventional memory to keep dos4gw.exe from running. When this happens, you will see "not enough memory" type error messages when attempting to run the extended macro functions. What to do? Three approaches:

- use DOS 5.0 or higher, and make sure the DOS=HIGH,UMB command is in your config.sys file
- run pminfo.exe and rminfo.exe, included with dos4gw.exe, to diag nose the current memory state and protected-mode compatibility of your machine, and make sure there's not a basic problem with switching into protected mode

- start "trimming" your system: REM-out config.sys and autoexec.bat command lines such as FASTOPEN and SMARTDRV, and remove other memory-resident utilities that take up conventional memory

Note that even for DOS 3.3x through 4.0x this is not a problem when running Hypersignal under DSPower Block-Diagram under Windows, because the met_shel.exe graphics driver is not loaded in this case, thus freeing up an additional 120k of conventional memory.

2.7 Windows Operation

If Hypersignal is executed from Windows, then it will be running in a "DOS box", which simulates a stand-alone DOS environment. Actually, however, the host machine is still in protected-mode (enhanced 386 or 486 operation), and "time-slices" are still being given to Windows, even though Windows is in the background (not visible). Thus, the potential exists for Hypersignal, and all its functions, displays, and hardware-based instrumentation, to run slower than if truly stand-alone. To minimize slowdown under Windows, take these steps:

- install Hypersignal-Macro or Hypersignal-Acoustic with an icon in a group of your choosing in the Program Manager. Set the command line to "hsm.bat" or "hsa.bat" (or "hs.bat"), as appropriate. This will cause hsm.pif or hsa.pif files included with Hypersignal to be used. These files are set to optimize performance under Windows. Also, when creating the icon, you can select Change Icon, then Browse, and then select the hsa.ico or hsm.ico icon file that was installed with Hypersignal. NOTE: when installing Hypersignal with the Signalogic Software Setup Utility, which is included with Hypersignal and DSPower software CDs, these steps are performed automatically.

- decrease time-slices given to Windows when in background mode to the minimum value of 1. To do this, go to Program Manager, select Control Panel (inside Main), and select 386 Enhanced.

Even with the above measures in place, you can still expect about a 10% slowdown in Hypersignal performance when running under Windows (in protected-mode) as opposed to stand-alone (real-mode). This is the nature of protected mode, and is true for all DOS programs.

NOTE: When Hypersignal is being used as a DSP Engine under DSPower, DSPower takes additional measures to ensure fast Hypersignal operation.

Display Drivers

Display drivers used in the Windows environment sometimes can cause problems. Basically, we have found the Windows VGA driver to be reliable, and allow use of Hypersignal either from a DOS shell commandline prompt, or from a "DOS launch" icon without problems. However,use of "SuperVGA" drivers, or 800x600 and 1024x768 drivers provided by the graphics board manufacturer sometimes cause problems. First, we have found that the manufacturer's drivers are almost always better than the general Windows "SuperVGA" driver. Second, when using Hypersignal, sometimes the background will not be fully erased in a display or instrument. In these cases, try the 'Refresh' command(ctrl-L) from the keyboard. Versions 4.00bx and later of Hypersignal contain special "force clear" code for some graphics boards that have demonstrated this problem. Third, any incompatibility or conflict arising from alternating use of MetaWindow DOS video drivers used by Hypersignal and Windows video drivers is avoided when Hypersignal is used with DSPower; in this case the MetaWindow drivers are not loaded.

Hypersignal .PIF Files

hsm.pif and hsa.pif files are included with Hypersignal-Macro and Hypersignal-Acoustic. These files are set to optimize performance of Hypersignal when launched from Windows (see Running Under Windows, above). However, if you wish to create your own .PIF files, the following settings are recommended:

- operation set to full-screen
- emulate text mode
- start in text mode (video memory)
- desire at least 384 kbyte of memory (maximum used is 640k)
- use high memory
- close window on exit
- no XMS, EMS, or locked memory is needed

- priority and background vs. exclusive operation are not important
- detect Idle Mode should be turned off (*very important for performance*)

When using Hypersignal with DSPower, DSPower uses prepared .PIF files that are substantially different from these settings, and should not be edited or changed under any circumstances.

Hypersignal Icons

.ICO files are included with HS-Macro and HS-Acoustic for use under windows (CD or distribution diskette 1). When setting up a Windows icon for these programs, install the program filename to 'C:\hsacous\hsa.bat',C:\hsmacro\hsm.bat', or similar--depending on the program and where it is located on disk. Then, select Change Icon, Browse, select the Hypersignal subdirectory, and then select the hsm.ico or hsa.ico icon files. Upon return to the Program Manager top-level, you should see the new Hypersignal icon. After double-clicking, Hypersignal should run full-screen, with brisk performance, and all waveform displays and other graphics displays should be full-screen and fast. If this is not the case, then make sure that the program filename for the icon (return to Properties) specifies the hsm.bat, hsa.bat, or hs.bat (HS-Macro NRT) batch files. This is very important, because use of these names causes hsm.pif or hsa.pif (or hs.pif) batch files to be invoked automatically.

NOTE: Signalogic's Windows-based Software Installation Utility, which is included on Hypersignal and DSPower software Cds, has setup options for installing a number of Signalogic software programs under Windows, and performs Hypersignal icon and .pif file configuration automatically.

this page intentionally left blank

3.0 USING HYPERSIGNAL

Although this chapter and chapter 4 both start with "Using Hypersignal...", this chapter covers the general Hypersignal user interface, while chapter 4 talks about using *specific* Hypersignal DSP and display functions. Keyboard and mouse interface, on-line Help, the main menu, the data-entry editor, invoking external programs, invoking macros, and finding waveforms are the basic subjects covered in this chapter.

This chapter assumes that you've already installed the Hypersignal software according to the instructions in section 2.1, Installing Hypersignal. Assuming that your software is up and running, topics covered here include executing Hypersignal with and without command-line parameters, some initial setup procedures once the software is running, and the basic, user-interface and system-interface functions available in the software (such as keyboard and mouse interface, on-line Help, the data-entry editor, printing screens and displays, etc.).

3.0.1 Advanced Features When Starting Hypersignal

From Windows clicks: Start, Program, Signalogic DSP , Hypersignal Macro icon. To start Hypersignal software, first path to the hard disk drive and subdirectory containing the Hypersignal executable code. Then, invoke one of the following batch files (which one you have depends on your software version) by typing:

HS	<ENTER>	(Hypersignal-Macro Non-Real-Time)
HSM	<ENTER>	(Hypersignal-Macro)
HSA	<ENTER>	(Hypersignal-Acoustic)
HSN	<ENTER>	(Hypersignal-Macro NRT, Nihon Version)
HSMN	<ENTER>	(Hypersignal-Macro, Nihon Version)
HSAN	<ENTER>	(Hypersignal-Acoustic, Nihon Version)

The batch file first invokes and makes memory-resident the Metagraphics graphics driver (described and explained in section 2.4, Selecting Graphics and Mouse Drivers, above), then runs Hypersignal. When you exit Hypersignal, the batch file removes the graphics driver from your system memory. This keeps the graphics driver out of your system memory when you are not using Hypersignal.

Automatic detection and usage of an 80x87 math coprocessor is performed every time the software executes. For more details on this process, and overriding it if necessary, see section 2.3, Using a Math Coprocessor.

Upon executing Hypersignal software for the first time, a message of the form "cannot find configuration file...Initializing new configuration" should be displayed. This message only appears when executing the software for the first time, because normally the user-environment configuration save file "HSMACRO.CNF" will be present on disk as a result of performing an "exit" function after a previous Hypersignal session. However, when running Hypersignal for the first time, no configuration file will be present—forcing you to specify graphics and mouse driver selections, since these cannot be determined automatically with 100% accuracy by the software. Please see section 2.4, Selecting Graphics and Mouse Drivers, for detailed information on choosing driver selections.

NOTE: If a configuration file is not found initially, then before printing the "...not found..." message Hypersignal first looks elsewhere, including the DOS path list, for a compatible configuration file. If one is found, then the software will start with this file, but save all new settings in a new file on the current path. This provides the means to set up a multiuser system with multiple configuration files: the initial configuration file can be provided from a master path, and individual, personalized configuration files saved to local paths. Also, since Hypersignal can be invoked from a distant path (see note below), a multiuser system can be further enhanced to operate with only one copy of Hypersignal located on the master path.

For advanced users, the following command-line options exist for Hypersignal execution:

EMS Memory Overlay

A '/e' entered immediately after the batch file name will cause Hypersignal to load its overlay file into EMS memory. However, this may limit the resolution of subsequent publication-quality prints—see section 4.3.2, Printing Waveform Displays.

Engine Mode

An '/e' entered as a command-line parameter will invoke "engine mode" operation , similar to driver mode (see below), but allowing Hypersignal to be invoked from DOS or Windows programs without a device driver being installed. As with driver mode, the standard Hypersignal graphics drivers are not loaded (freeing up about 130 kb of low memory), which causes display and instrument functions to be disabled. Also like driver mode, Hypersignal runs "quietly" and produces no screen output of any type. Below are DOS command line and Windows C/C++ program examples to do this:

```
hsmacro /e fname.mac

hsacous /e fname.mac

StatusCode = WinExec("C:\\hsm\\hsmacro.pif /e fname.mac", SW_HIDE);
StatusCode = WinExec("C:\\hsa\\hsacous.pif /e fname.mac", SW_HIDE);
```

In engine mode, Hypersignal interrogates the system for an external device "pipe" driver, and if one is not found (or is not connected to another application), disables interactive communication, graphical output, error/status output, and all macro language non-file input/output. The macro language file specified by fname.mac should contain no display or instrument functions. Also, it should not use the $MANUAL or $WAITFORKEY commands, and it should include an Abort or Exit command to cause termination of the session. Note that the HSxxxx.bat files should not be used from the DOS command line or WinExec routine, or the graphics drivers will be installed unnecessarily.

Driver Mode

A '/d' "driver mode" flag can be invoked on the command line which will
cause HS-Macro to invoke no graphics displays, and make no keyboard inquiries—avoiding user I/O hardware accesses altogether. This flag should be specified only when HS-Macro has been remotely invoked from an external DOS or Windows application (using a device driver to talk to Hypersignal), and has been instructed to run a macro language file at startup. The macro language file should contain no displays that use the $MANUAL or $WAITFORKEY commands. Below is an example of a WinExec call to do this:

```
StatusCode = WinExec("C:\\hsm\\hsmacro.pif /d fname.mac", SW_HIDE);
```

A device driver such as pipe.386 (included with DSPower) software should be installed in the system prior to using this mode. The HSMACRO.PIF file should be set up as a windowed application, 384 kb required, and background operation; I/O ports and memory need not be monitored. No memory needs to be locked. Be sure that fname.mac includes an Abort or Exit command in order to terminate the process. Also, do not invoke the HSM.BAT file, as this will install the MetaWINDOW graphics driver, which is not needed.

Board Address Override

Next, for university program packages or other special versions of the software that require specific hardware in the system in order to operate, a DSP/Acquisition board address can be entered in the form '/$HHH', where HHH is a hexadecimal value.

Intitial Macro Program

Next, a macro language filename can be entered in the form 'filename.mac'. If filename.mac specifies a valid macro source code file, then the software will immediately execute the macro file upon executing. This provides a method for executing Hypersignal functions remotely, from the operating system or from another program. Input parameters with which the software will initialize the Macro Manager menu before executing the macro can be specified after the macro filename by preceding them with a '#' character. For example, you could instruct Hypersignal-Macro to run and then immediately compile (if necessary) and execute the "SAMCONV.MAC" file included on distribution diskette #2 with the following command-line statement:

```
HSM samconv #speech #9 #4 #sp_out  <ENTER>
```

This will initialize the input waveform to "speech", the stretch and contract integers to 9 and 4, respectively, and the output waveform to "sp_out". Up to 6 input parameters can be placed in the command line. Note that the macro language file can use the "Exit" or "Abort" commands to automatically leave Hypersignal upon completion of the macro file, relinquising control if necessary; see section 7.10, Macro Commands and Functions, for more information. You could even specify an input parameter that the macro could use to *conditionally* determine whether to exit automatically, in case the macro was also one used frequently from inside Hypersignal, as in the following macro source code example:

```
if #3 = "Exit" then
    Exit                    ; save config file and exit
endif
if #3 ="Abort" then
    Abort                   ; exit without saving config file
endif
```

In the above example, Hypersignal will terminate depending on the value of input parameter #3 passed in the command line. However, if you needed to execute the macro from inside Hypersignal without exiting, input parameter #3 could entered in the Macro Manager menu as anything but "Exit" or "Abort" to prevent the macro from causing Hypersignal to terminate.

Finally, the TIME and FREQ FILENAME fields in the User Setup area can be initialized to default filenames.

Below is an example of invoking Hypersignal-Macro with several command-line parameters:

```
HSM /E demo.mac speech signal  <ENTER>
```

In the above example, the overlay file will be loaded into EMS memory, the macro file "demo.mac" will be executed immediately, the User Setup TIME FILENAME will be set to "speech", and the User Setup FREQ FILENAME will be set to "signal".

Below is an example of invoking a university version of Hypersignal-Acoustic with some command-line parameters:

```
HSA /$290 speech  <ENTER>
```

In the above example, the DSP board address is set to 290H, and the User Setup TIME and FREQ FILENAMEs will both be set to "speech".

NOTE: **You can invoke Hypersignal from a remote path. Generally, the software searches for system files, the on-line Help file, and printer-driver definition file first on its own path, then on the currently logged DOS path, and then in the DOS path list. The on-line Help and printer-driver definition files are also searched for on the path specified in the User Setup area.**

3.1 Keyboard and Mouse Interface

Below is listed the general function of keyboard keys and mouse buttons during the four basic types of Hypersignal displays: main menu, data-entry menu, waveform display, and on-line Help display. Only commands that are used frequently in *all* four major areas are listed here. For a complete list of data-entry editor, waveform display, and on-line Help commands, see sections 3.4, 4.3.1, and 3.2, respectively. Note that for two-button mice, functions listed as responding to the middle button, "two leftmost" buttons, or "two rightmost" buttons will be deactivated. For more detailed information on individual keyboard and

mouse functions, please see the appropriate section in this manual pertaining to the display/DSP function you are concerned with.

	main menu:	data-entry menu:	waveform display:	on-line Help:
'Enter' key:	select currently highlighted menu selection	move down one field; advance to processing and/or display function if cursor is on last field	reset waveform display to initial settings	page forwards in on-line Help text
left,right,up, down arrow keys, mouse motion:	highlight menu entree	move editing cursor	move target window or move crosshair cursor, depending on mode	page forwards and backwards inside on-line Help text
'Home' key, left mouse button:	select menu entree	moves cursor down one field (left mouse button); moves cursor to start of field ('Home' key)	zoom in on waveform if in "region" mode, attach crosshair cursor to waveform if in "point" mode	jump to start of Help text
spacebar key, right mouse button:	select menu entree	no action (spacebar), moves cursor down one field (right mouse button)	pan forward to next waveform display frame	jump to end of Help text (right mouse button only)
middle mouse button:	select menu entree	moves cursor down one field	pan backwards to previous waveform display frame	page forwards in on-line Help text
'Backspace' key:	no action	backup and delete character	pan backwards to previous waveform display frame	page backwards in on-line Help text
'Esc' key, leftmost two mouse buttons:	no action	return to main menu	return to data-entry menu	return to main menu, data-entry menu, or waveform display
'F1' key, rightmost two mouse buttons:	display on-line Help for currently highlighted menu selection	display on-line Help for current processing/display function	display on-line Help for current processing/display function	invoke general Help, text
outer two mouse buttons together:	no action	no action	zoom-out for waveform display ("zoom undo")	no action
'F3' key:	invoke Macro Manager menu	no action	no action	no action
'F4' key:	invoke Macro Record menu	no action	no action	no action
'F10' key:	invoke DOS shell	no action	no action	no action
'ctrl-Enter' key sequence (or 'ctrl-J' on NEC PC-980x machines):	no action	advance to processing and/or display function	no action	no action
'End' key:	no action	moves cursor to end of field	no action	jump to end of Help text

'Insert' key:	no action	put data-entry editor into "insert" mode	insert samples into waveform if in "region" mode for those display functions where editing is allowed	no action
'Delete' key:	no action	delete character at current cursor position	delete samples from waveform if in "region" mode for those display functions where editing is allowed	no action
'PgUp' key:	no action	initiates "pop-up directory" display	scale waveform up	page backwards in on-line Help text
'PgDn' key:	no action	shows next page of "pop-up directory" display	scale waveform down	page forwards in on-line Help text

3.2 On-Line Help

On-line Help may be accessed from anywhere in the Hypersignal program by using the 'F1' key, or the rightmost two buttons on a three-button mouse. From the main menu and inside waveform displays, the '?' and 'H' keys also access on-line Help. During waveform displays, always use these keys and/or mouse bottons, even though legends may not always be shown during waveform displays.

Within any Help text display, accessing Help again will begin a general Help text display; the general Help text is always available even if the on-line Help file ("HSMACRO.HLP") should be lost or destroyed.

The number of pages in the Help display will vary between menu selections. Use the 'PgDn', down arrow keys, or move the mouse down to page forward, and the 'PgUp', up arrow keys, or move the mouse up to page backward. The 'End' key (or righthand mouse button) will jump to the end of the displayed Help text. The 'Home' key (or lefthand mouse button) will jump to the start of the displayed Help text. You may return to the previous menu at any time by pressing the 'Esc' key (or the leftmost two buttons on a three-button mouse).

Hypersignal searches the currently logged DOS path, the path containing the "HSxxxxx.EXE" file (DOS 3.3 and above only), the User Setup path, and the DOS path list for the on-line Help file. If not found, Hypersignal then searches the current subdirectory and logged drive shown in the User Setup area. If still not found, Hypersignal will report the Help file not found and disable on-line Help display during the current session until such time as the current subdirectory and logged drive in the user setup area is changed to correctly access the on-line Help file. General Help is always available, since it is imbedded in the main program file, "HSxxxxx.EXE". If you should accidently damage or destroy the on-line Help file, then contact your distributor for a replacement.

3.3 The Hypersignal Main Menu

After starting the Hypersignal program as outlined in previous sections, the Hypersignal main menu similar to one will be displayed. Once the main menu is displayed, the following keys are active (note that most of these key functions are listed horizontally at the bottom of the main menu screen):

'Q'	Quits Hypersignal ***without saving or updating the user-environment configuration file***
'E'	Exits Hypersignal and ***saves/updates the user-environment configuration file***
'U'	***Saves/updates the user-environment configuration file*** without exiting Hypersignal
'F1', '?', 'H', two right most buttons on three-button mouse	Begins on-line Help display for current menu function selected (highlighted selection)

'F3'	Invokes the Macro Manager menu, in which a macro source file can be specified for compilation and/or execution (see section 3.10, Invoking Macros)
'F4'	Invokes the Macro Record menu, in which an empty macro source file can be specified to which to record manual function and display selections (and display commands) for later "playback" as a macro (see section 4.10.2, Using the "Record" Mode to Create Macros)
'F10'	This is the DOS "Hot-Key"—it activates a temporary DOS shell in which standard DOS commands can be entered and executed (type 'Exit' to return to Hypersignal)
Arrow keys or mouse motion	Change current (highlighted) menu selection in respective direction (up, down, left, or right)
'Enter', spacebar, any mouse button	Selects currently highlighted menu function

The main menu can be divided into three basic areas. The largest, occupying the lower 2/3 of the screen, is the menu selection area. In this area, the arrow keys or mouse can be used to change the current menu highlight, and the 'Enter' (carriage return) key or any mouse button can be used to select a particular function.

At the upper right of the screen is the User Setup area. This area may be accessed by the User Setup menu selection, under the Utilities column heading, at the far right of menu selection area. In this area, the current logged drive and subdirectory, currently selected time and frequency domain filenames, and several global program parameters may be modified. See section 3.3.3, The User Setup Area, for more detailed information on this part of the main menu.

At the upper left of the screen is the System Status area. This area cannot be accessed by the user. Status parameters include the current time and date, running Hypersignal session time, and current menu function selected. The current menu function selected will display "MAIN MENU" until menu selection, after which it will display the appropriate menu label. The System Status area is also used by the data-entry editor as a "Waveform Summary" display area when waveforms have been loaded, and for a "Pop-up Waveform Directory" display area when the program cannot locate a specified waveform filename (see sections 3.4.1 and 3.4.2). See section 3.3.4, The System Status Area, for detailed information on this part of the main menu.

3.3.1 Menu Selection

All Hypersignal functions can be highlighted in the main menu using mouse motion or the arrow keys and then selected with the 'Enter' (carriage return) key or any mouse button. There are three categories of Hypersignal functions, which are divided into columns in the main menu display: Time Domain, Frequency Domain, and Utilities. Appendix B, Hypersignal Environment Flow Diagram, shows a graphical representation of all menu selections and waveform displays, and what type of waveform data they use as input and create as output, if any. Pressing the 'Enter' key, space bar, or any mouse button will invoke the currently highlighted menu selection.

Once inside the "data-entry menu" for any menu selection (see section 3.4, Using the Hypersignal Data-Entry Editor, below), pressing the 'Enter' key (or any mouse button) with the cursor on the last data-entry field, or pressing the 'ctrl-Enter' key sequence at any time advances to function processing or graphical display, as the case may be.

At any time, you may access on-line Help for the currently highlighted menu selection. Use the '?', or 'F1' keys to display of relevant Help information. *The 'Esc' (escape) key always returns to the main menu.*

3.3.2 To "Quit" Or To "Exit" Hypersignal

You can only leave Hypersignal from the main menu. You can do this two ways: Quit ('Q' key), or Exit ('E' key). If you quit, Hypersignal stops everything on the spot, the current Hypersignal user-environment is not saved, and the operating system prompt returns immediately. If you exit, Hypersignal saves the current user environment, including graphics and acquisition hardware driver selections, all current filename selections, display options, menu selections, and so forth, in the user-environment configuration file ("HSMACRO.CNF") and then returns to the operating system prompt.

When you leave Hypersignal, the current DOS path will be the same as when you started Hypersignal, unless you shelled out to DOS or to an external program during Hypersignal execution and changed the logged path with the RESTORE DOS PATH

field in the External Program Configuration menu disabled (see section 3.6, Invoking External Programs and the DOS "Hot-Key").

3.3.3 The User Setup Area

When menus are displayed (screens other than graphical displays), the upper right of the screen will reflect the current User Setup. This area may be accessed by the User Setup menu selection, under the Utilities column heading (see section 2.5.1, Initializing the User Setup Area). In this area, such parameters as the current logged drive and subdirectory, currently selected time and frequency domain waveform filenames, dual display autoscaling and global program sound enable/disable may be modified.

The logged drive and subdirectory parameters apply to *all Hypersignal waveform files.* The current TIME FILENAME and FREQ FILENAME can be used to initialize the functions under the Time Domain and Frequency Domain column headings, respectively. In practice, setting these values via the User Setup menu item is of little worth, because all filenames can be set in the individual functions anyway, and the program does such a good job of automatically tracking waveform names from one function to another.

All user setup parameters are initialized from the user-environment configuration file, including the current drive and subdirectory. If a configuration file cannot be found, then the User Setup parameters will be initialized to a default state (drive and subdirectory will be initialized to their DOS values at the time of program start).

Should subdirectory information need be entered that is longer than the subdirectory field in the system setup area, additional subdirectory information may be entered following the drive identifier (drive plus colon) in the drive field. When accessing files, Hypersignal forms the search path by concatenating the drive and subdirectory fields, *exactly as they are shown onscreen in the User Setup area.* If using both fields for subdirectory information, be careful that consecutive or extra forward slashes do not occur.

If you need to specify a path longer than the DRIVE and SUBDIRECTORY fields combined, then set this path directly in the user-environment configuration file. Even though the path will be truncated for display purposes on the main menu, its internal length will remain intact. In such a case, do not edit the path directly in the program. If you do, its internal value will be truncated also.

3.3.3.1 Dual Autoscaling

The DUAL AUTOSCALE field in the User Setup area is important enough to warrant further discussion in this section. This field disables or enables automatic scaling by the program during dual waveform displays. When enabled (an entry of 'ON'), the program automatically scales the smaller of two waveforms in a dual display to so that its maximum absolute amplitude matches the larger. When disabled (an entry of 'OFF'), no scaling is performed, allowing waveform amplitudes to be compared relatively. The default value for this field is 'FILE', in which case the program automatically enables autoscaling for dual displays where one or both waveforms have the "AUTOSCALE" bit in their waveform headers enabled, and disables autoscaling in all other cases. Hypersignal does not currently set the AUTOSCALE bit for any reason, but this bit can easily be set by user-defined external programs or macro language processing. See chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT, for more information about the AUTOSCALE bit and waveform headers.

3.3.4 The System Status Area

When menus or data-entry screens are displayed (screens other than graphical displays), the upper left of the screen shows the "System Status". This area cannot be accessed by the user. System Status area parameters include the time and date, running Hypersignal session time, and current function selected.

The current function selected will display "MAIN MENU" while the main menu is displayed, and indicate the current function after selection from the main menu.

The time and date are initialized at program start, and updated with normal program use thereafter.

The running session time indicates the length of time Hypersignal has been in use continuously. It is also updated with normal program use. If no clock is active in the host computer, then the time and date will be correct only if they were set manually at the time of the DOS prompt during computer power-on sequence.

The System Status area is also used to display various other information during the course of normal program operation:

STATUS MESSAGE DISPLAY

Various Hypersignal status message and non-critical error conditions may be reported in the System Status area from time to time. See section 3.3.4.1 below, Status Message Display, for more information.

CRITICAL ERROR MESSAGE DISPLAY

Various DOS and Hypersignal critical error conditions may be reported in the System Status area from time to time. See section 3.3.4.2 below, Critical Error Message Display, for more information.

WAVEFORM SUMMARY DISPLAY

A summary of the contents of waveform file headers is displayed upon successful load of a waveform for subsequent display or processing. See section 3.4.1, Waveform Summary Display, for details.

POP-UP WAVEFORM DIRECTORY DISPLAY

When a specified waveform filename can't be located, or upon user request, the System Status area is used to show pop-up directory information based on the current processing/display function being executed. A waveform filename search strategy is employed to allow the user to search through likely categories of waveforms depending on what the required waveform file types are in the current funciton. See section 3.4.2 below, Pop-Up Directory Display, for a detailed description.

3.3.4.1 Status Message Display

Non-critical Hypersignal-detected error or status conditions, such as a user-defined program not found or lack of disk space to perform an "Exit" function (and save the user-environment configuration file) are displayed in the System Status area. Also, the macro language facility uses this area extensively, displaying all compilation and execution status or error messages in the System Status area.

3.3.4.2 Critical Error Message Display

Critical errors, including DOS-related error conditions such as write-protected diskettes, drive not ready, and media error (bad sector, seek error, etc.) are displayed in the System Status area at the upper left of the main menu. All DOS error messages that would normally overwrite the current display (including the infamous "Abort, Retry, or Ignore?" messages) are intercepted by Hypersignal, translated, and displayed (if necessary) in the System Status area along with the disk affected, if any.

3.4 Using the Hypersignal Data-Entry Editor

The Hypersignal data-entry editor is used to allow convenient entry and modification of parameters needed to execute Hypersignal processsing and display functions.

Modifiable screen data for all data-entry menus are arranged as a column of "fields". A field is a specific place on the screen where data may be entered or changed. Each field consists of one or more characters. Characters may be any valid keyboard character except for the spacebar, unless otherwise noted. The current character position in the current field being edited is indicated by a blinking cursor. A large blinking cursor is used when in insert mode (enabled by the 'Insert' key; see below), and the standard DOS cursor is used when in Overwrite mode. A field is "entered" by the 'Enter' (carriage return) key or a mouse button, after which the cursor advances to the next field automatically. The arrow keys or mouse motion can be used to move the cursor from character to character and from field to field.

There are two ways to advance to actual function processing or graphical display: 1) entering the last field (pressing the 'Enter' key or any mouse button when the cursor is on the last field in the menu), or 2) pressing the <CTRL-ENTER> key sequence (use <CTRL-J> key sequence on NEC PC-980x machines). Doing either of these will cause function processing or display to begin, as appropriate.

All fields in a data-entry menu except the last field may be revised as many times as necessary by repositioning the cursor to that field and entering the field.

At any time, you may access on-line Help for the currently invoked processing or display function. Use the 'F1' key to begin display of relevant Help information. The 'ESC' (escape) key will return to the data-entry menu.

To return to the main menu from any data-entry menu, use the 'ESC' (escape) key.

The data-entry editor has two basic modes, controlled through the EDITOR UNDO field in the User Setup menu selection. The default value for this field is 'OFF', which will cause the up and down arrows or vertical mouse motion to keep any changes made in the previous field. An 'ON' entry for the EDITOR UNDO field will cause the up and down arrow keys and vertical mouse motion to "undo" any changes on the previous field—allowing only the 'Enter' key or mouse button to finalize a field.

When a filename field is entered, data beyond the cursor position is cleared. This does not occur for fields which contain "subfields", or two or more separate entries separated by commas, since it may only be one subfield of the field that needed changing. This also does not occur in almost all cases if the cursor is at the very beginning of a field, regardless of the type of field.

After entering the Insert mode (see 'INSERT' key, below), any type of entry that causes a field change (up and down arrow keys, mouse motion, 'ENTER' key) automatically disables the Insert mode.

Following are the active keyboard controls when editing Hypersignal function data entry displays:

Arrow keys or Mouse motion	All arrow keys and mouse movement are active in the editor, allowing you to move within and between fields without changing data. Arrow keys will "wrap-around" when at the first and last fields up and down, or the first and last characters left and right. Mouse motion never causes a wrap-around, because it is so easy to overmove the mouse.
'Enter' key (carriage return) or any mouse button	Used to "enter" a field. Whatever data presently contained in a field will be finalized when the 'Enter'key or a mouse button is pressed. A field may be entered regardless of the current cursor position in the field. After entering the last field in a data-entry menu, function processing or graphical display, as appropriate, will begin.
<CTRL-ENTER> key sequence (or <Ctrl-J> for NEC PC-980x machines)	"Enters" the current field and causes function processing or graphical display to begin.
'DELETE' key	Deletes the character in the current cursor position without changing cursor position. All characters to the right of the cursor are moved to the left one space.
'INSERT' key	Causes the data-entry editor to enter Insert mode, in which new characters do not overwrite characters already entered, but instead are inserted at the current cursor location. Insert mode is indicated onscreen by a large blinking cursor. Pressing the 'Insert' key again returns to Overwrite mode, and also returns the cursor to its original, standard DOS size.
'BACKSPACE' key	Deletes the character immediately to the left of the current cursor position, and then moves to the left one space. All characters to the right of the cursor are moved to the left one space.
'ESC' key (or leftmost two buttons on three-button mouse)	Returns to the main menu. Any fields already entered remain so.
'PGUP' key	Begins pop-up directory display for the current function, if not already started because the program couldn't locate a specified waveform filename. The waveform filename and extension used depends on the current function and/or menu selection. Consecutive use of the 'PgUp' key causes the pop-up directory display to first return to the first page of the current directory display (if not on the first page), and then starts changing the current filename and extension search strategy in use. See section 3.4.2 below, Pop-Up Directory Display, for details.

'PGDN' key	Used to display the next page of pop-up directory information. Pop-up directory information is displayed during functions requiring input waveform filename entry when the input filename cannot be found, or in response to the 'PgUp' key, above. See section 3.4.2 below, Pop-Up Directory Display, for details.
'HOME' key	Moves cursor to beginning of current field.
'END' key	Moves cursor to end of current field.
'F1' key (or rightmost two buttons on three-button mouse)	Causes on-line Help text for the currently invoked processing or display function to be displayed. The 'Esc' (escape) key will return to the data-entry menu.

3.4.1 Waveform Summary Display

Waveform summaries are displayed in the System Status area (upper left of main menu) after successful loading of waveform file(s) for display or processing. A waveform summary displays parameters and attributes from the waveform file header (see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT), including the number of samples in the waveform, type of data (arbitrary time waveform, LPC autocorrelated, filter construction output, etc.), whether or not the time domain file is real or complex, number of channels, maximum amplitude, sampling frequency, FFT order, and time domain analysis framesize.

To redisplay a waveform summary that has been overwritten by pop-up directory display information or error messages, you can reload the waveform by repositioning the cursor at the beginning of the waveform filename field and pressing the 'Enter' (carriage return) key.

In the case of dual-waveform display functions, the waveform summary always applies to FILENAME1.

Below is an example of a waveform display summary:

```
Number of samples:    1015672
Type/complex/numchn:  magH[n]/real/1
Max |amplitude|:      19103.0
Analysis framesize:   4096
FFT order:            12
Sampling frequency:   44100Hz
```

Especially note the "Type/complex/numchn" descriptor, which, in this example, indicates a non-complex, single-channel, filter (transfer function, or "H[n]") magnitude (frequency domain) waveform file. The "type" of a waveform actually consists of two parts: the type of data, and its original source. Currently, the type of waveform data can be "tim" for time domain data, "mag" for frequency domain magnitude data, "phz" for frequency domain phase data, or "pwr" for frequency domain power spectra data. The source of data can be "" (null) for arbitrary time domain data (e.g. A/D input), "H[n]" for filter construction or other transfer function output, or "LPC" for LPC Autocorrelation output. The "complex" descriptor can be "real" for data that was non-complex in its original time domain form, or "cplx" for data that was complex. Complex data exists when at least one imaginary component in a time waveform file is non-zero. See chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT, for a description of the complex attribute of waveform file headers and the format of both complex and real data.

Below is a second example of a waveform display summary:

```
Number of samples:    10000
Type/complex/numchn:  tim/cplx/2
Max |amplitude|:      31596.25
Analysis framesize:   144
FFT order:            9
Sampling frequency:   1568100Hz
```

The above example is for a complex, two-channel, arbitrary time domain waveform file.

NOTE: With version 3.10 or later of Hypersignal-Macro, the "Max |amplitude|" value that is displayed can be a floating-point value. For a full explanation of waveform ampltitude values and waveform header exponents, see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT.

3.4.2 Pop-Up Directory Display

Pop-up directory display appears during data-entry editor menus as an automatic window display in the System Status area (upper left of main menu) when a specified filename cannot be found. Also, you can manually invoke pop-up directory display from inside any data-entry menu by using the 'PgUp' key.

Pop-up directory display includes waveform filename and extension, length, and date of creation. The search strategy for different waveform extensions and/or more specific waveform filename may be changed via consecutive use of the 'PgUp' key (see section 3.4.2.1 below, Pop-Up Directory Waveform File Search Strategy).

During time domain, frequency domain, and some utility functions requiring input waveform files, if a waveform filename is not found after being entered by the user, then candidate filenames having appropriate filename extensions will "pop-up" in the System Status area. The initial filename extension used is based on the current function. For example, when performing an FFT function, an input filename not found will prompt display of those filenames on the current path with ".TIM" extensions, since the FFT is a time domain function. Other time domain extensions include ".FFD" and ".IFD" for Filter Construction filenames, ".IIR" for Recursive Filtering input filter coefficient files, and ".EQU" for Difference Equation save files. Continued use of the PgUp key will change the search path to "alternate" filename extensions, or a "filename.∗" search; see section 3.4.2.1 below, Pop-Up Directory Waveform File Search Strategy.

During frequency domain functions requiring filename entry initial filename extensions used for pop-up directory display can be ".FRQ" (magnitude), ".PHZ" (phase), ".IMA" (imaginary), ".REA" (real), or ".PWR" (power spectra).

For utilities selections the value for the extension varies, depending on menu selection. For External Program Configuration, the extensions are ".COM", ".EXE", and ".BAT". For ASCII Export, the extension is ".TXT". For File Acquisition and ASCII import, the extension used is the extension currently entered as part of the input filename field of the data-entry menu, or the last valid one set for time domain or frequency domain functions. This is also true for all other menu selections that do not require filename entry: they use the last valid extension set by selections that do require filename entry.

3.4.2.1 Pop-Up Directory Waveform File Search Strategy

Consecutive use of the 'PgUp' key will change the search stategy for the current pop-up directory display. After initial display, in which the entered filename has been replaced by a "∗" and an appropriate filename extension added, a second 'PgUp' key will cause the currently entered waveform filename and extension to be used in the directory search. If no extension is entered, a ".∗" extension will be added to the entered filename. Thus, the second 'PgUp' key effectively shows all occurrences of the filename you've entered, regardless of extension. A third 'PgUp' key will use the currently entered filename along with an "alternate extension"; for example, filter design functions have a primary extension of ".IFD" or ".FFD" because of the specification files they use, but they also have an alternate extension of ".TIM", because they are time domain functions. As another example, Difference Equations have a ".EQU" primary extension and a ".TIM" alternate extension. If no alternate extension applies to the function selected, a ".∗" extension is substituted. Further 'PgUp' keys cause the process to start over.

You may also temporarily enter "∗" or "?" wildcard characters after or inside the filename to further direct a waveform search; for example a 'test∗' entry would include all filenames starting with "test", and having the appropriate extension. As another example, an entry of 'x?test' would include all filenames starting with "x", ending with "test", and having any other character in between.

NOTE: **To re-initialize pop-up directory display after you have changed the filename in the data-entry menu, or added a wildcard character ('∗', '?'),** ***you must reposition the cursor at the beginning of the waveform filename field and 'ENTER' the field*****—then press the 'PGUP' key.**

3.5 Printing Menus and Waveform Displays

To print Hypersignal menu displays, you should use the standard DOS <SHIFT-PRINT SCRN> key sequence; there is no built-in Hypersignal function for printing menu displays.

To print waveform displays, both to printers and disk files, you should use the built-in Hypersignal function that can perform both "screen dumps" and "publication-quality" prints. A screen dump is defined as simply transferring the current contents of

video RAM directly to the printer, without any increase in resolution. Of course, for printers with resolution greater than about 90 dots per inch, this will not fully utilize the capability of the printer. A publication-quality print is defined as regenerating the current screen display in EMS memory, at much higher resolution, and then transferring this image to the printer.

The built-in waveform print function is invoked by the 'F2' key inside all waveform displays. See section 4.3.2, Printing Waveform Displays, for full details on using the built-in print function.

NOTE: If you need to add a printer to the printer-driver definition table (pdt), or modify an existing entry, then you can invoke the pdt editor. The file "BDE.EXE" has been provided for this purpose; by typing 'BDE' at the DOS command line (outside of Hypersignal), you will bring up a listing of all current printers. A special-purpose editing mode is available that allows you to set various printer specifications and parameters, and also to set special "escape sequence" commands for the printer. Follow the instructions in the on-line Help for when modifying or adding printer entries. If you need further assistance, please contact your distributor.

3.6 Invoking External Programs and the DOS "Hot-Key"

From the Hypersignal main menu, you can execute external programs, and also invoke a DOS "shell" (actually an executable program in the form of the DOS "COMMAND.COM" file).

To enhance this capability, "User-Defined" menu labels are available on the main menu (two each under the Time Domain, Frequency Domain, and Utilities columns). For example, you could set up a user-defined utility menu label to point to a text-editor, and time domain and frequency domain menu labels to point to your own DSP processing programs.

Also, the F10 key has been installed in the main menu as a DOS "Hot-Key", which allows you to jump to a DOS shell at any time from the main menu, perform DOS operations, and "exit" quickly back to Hypersignal.

NOTE: Hypersignal updates the user-environment configuration file before executing an external program (including DOS shell) and reads the configuration file back again after the external program finishes executing. In this way, a user-created external program can access and modify the configuration file, causing changes in the Hypersignal environment upon return. In order to guarantee that the configuration file can be read after the external program (or DOS shell) returns, Hypersignal saves the current DOS path (current logged drive and/or subdirectory) before yielding control, and restores the saved value afterwards. If you wish to override the automatic restoration, because external or macro processing needs to change DOS paths, you can set the RESTORE DOS PATH field in the External Program Configuration menu to 'N'. However, keep in mind that if you disable automatic restoration, and then change the current logged drive and/or subdirectory while in an external program or DOS shell, Hypersignal will read (or attempt to read) a new configuration file, if any, after the external program finishes.

Use the Ext Program Config menu selection to set up your external program filenames and desired Hypersignal menu labels. For example, a typical External Program Configuration menu might look like this:

```
TIME FILE,MENU LABEL1:   sinegen.exe,sinegen
TIME FILE,MENU LABEL2:   prog1.com,menu_lab_1
FREQ FILE,MENU LABEL1:   prog2.exe,menu_lab_2
FREQ FILE,MENU LABEL2:   prog3.xxx,menu_lab_3
UTIL FILE,MENU LABEL1:   edit.com,text-editor
UTIL FILE,MENU LABEL2:   prog5.yyy,menu_lab_5
                DRIVE:   C:
        SUBDIRECTORY:    HSIGNAL
```

where "sinegen.exe" is an example external program provided on the distribution diskettes (intended as an example of how to access the Hypersignal user-environment configuration file), "edit.com" is a prototypical text editor, and "progN.ext" are prototypical user-defined programs.

If not enough memory is available to run an external program from Hypersignal, see if you can load the "MET_SHEL.EXE" graphics driver into "high" memory, above the 640 kByte conventional memory. Whether this can be done or not is highly system-dependent. See section 2.4.3, Installing the Metagraphics Driver Above Conventional Memory, for more information.

Also, make sure that all other memory-resident device drivers not needed when running Hypersignal have been removed (or relocated to high memory).

NOTE: **You should not install memory-resident drivers from a DOS invoked during Hypersignal operation. Why? Because of the graphics driver ("MET_SHEL.EXE"), which is installed as memory-resident just before Hypersignal runs (done automatically by the HSx.BAT batch file), and is removed from memory just after Hypersignal runs. In order for "MET_SHEL.EXE" to be removed, it must be the *last memory-resident driver installed.* If not, because another driver has been installed after it, "MET_SHEL.EXE" cannot be removed and will continue to occupy system memory even after Hypersignal has finished executing.**

See section 2.5.3, Initializing the External Program Configuration Menu, for more information on setting up external programs and defining main menu labels.

3.7 Updating and Accessing the User-Environment Configuration File

The "user-environment configuration file" is an ASCII text file that contains the complete state of all user-controlled parameters in Hypersignal. This file is intended to be saved between Hypersignal sessions so that frequent use of the software is as convenient as possible, and no time is spent rethinking set-up or initialization parameters. The configuration file is saved on disk under the name "HSMACRO.CNF" for all versions of Hypersignal software.

The user-environment configuration file includes the current graphics and mouse hardware drivers, current System Configuration, currently selected acquisition/DSP hardware (real-time software only), current User Setup area, all current menu selections, all current user-modifiable data (filenames, analysis parameters, display options, etc.), current editing cursor position, and so forth. Every user-modifiable parameter in Hypersignal is saved in the configuration file.

The user-environment configuration file can be saved from Hypersignal in three ways:

1) By "exiting" from the main menu ('E' key). Exiting the software saves the configuration file and stops Hypersignal execution.

2) By "updating" from the main menu ('U' key). Updating saves the configuration file but Hypersignal continues to execute.

3) By executing an external program or DOS shell. The configuration file is saved automatically by the program in anticipation of access by the external program.

NOTE: **If you "quit" from the Hypersignal main menu, the user-environment configuration file is not saved.**

When Hypersignal next executes, or control is returned to Hypersignal, the user-environment is initialized to a state defined by the configuration file contents. See section appendix A, User-Environment Configuration File Format, and the "CONFIG.DOC" file included on the distribution diskettes, for a listing of configuration file contents.

Upon executing Hypersignal software for the first time, a message of the form "cannot find configuration file...Initializing new configuration" should be displayed. This message only appears when executing the software for the first time, because normally the configuration file will be present on disk as a result of performing an "exit" function after a previous Hypersignal session. However, when running Hypersignal for the first time, no configuration file will be present—forcing you to specify graphics and mouse driver selections, since these cannot be determined automatically with 100% accuracy by the software. Please see section 2.4, Selecting Graphics and Mouse Drivers, for detailed information on choosing driver selections.

NOTE: **Hypersignal checks for a "compatibility identifier" in the configuration save file before reading it. If you have received an updated version of software, there is a good chance your old configuration file won't be "compatible" (due to parameters being added at the end of the file), and the program will display a message of the form "...configuration file not compatible...initializing new configuration..." upon first executing. We apologize for any inconvenience caused by having to reset some choices and options.**

Below are some notes of interest about accessing the user-environment configuration file:

The example external programs on the distribution diskettes all "include" the "CONFIG.DOC" file as part of their source code. The "CONFIG.DOC" file contains an example procedure that searches for the configuration file, opens the file, reads in all parameters, changes a few parameters (by way of example), writes all parameters back to the file, then closes the file.

The current DOS path (drive and subdirectory) is the first place where Hypersignal expects to find the configuration file. Moreover, this is *always* the place where Hypersignal saves its configuration file. If Hypersignal cannot locate a configuration file on the current DOS path, it then searches the path containing the executable file ("HSxxxxx.EXE") and the DOS path list. If a compatible configuration file is found, then the software will start with this file, but save all new settings in a new file on the current path. Note that this provides the means to set up a multiuser system with multiple configuration files: the initial configuration file can be provided from a master path, and individual, personalized configuration files can be saved to local paths.

Because the current DOS path is used as the default location for the configuration file, this value is not saved inside the configuration file itself. Please do not confuse the current DOS path with the User Setup path (DRIVE and SUBDIRECTORY fields at upper right of main menu). The User Setup path is used to locate all Hypersignal waveforms, and is, of course, saved inside the configuration file. Thus, changing the User Setup path (or the External Program Configuration path or the Macro Manager path or any other path specifiable inside Hypersignal) has no effect on the location of the configuration file.

Hypersignal updates the user-environment configuration file before executing an external program (including DOS shell) and reads the configuration file back again after the external program finishes executing. In this way, a user-created external program can access and modify the configuration file, causing changes in the Hypersignal environment upon return. In order to guarantee that the configuration file can be read after the external program (or DOS shell) returns, Hypersignal saves the current DOS path (current logged drive and/or subdirectory) before yielding control, and restores the saved value afterwards. If you wish to override the automatic restoration, because external or macro processing needs to change DOS paths, you can set the RESTORE DOS PATH field in the External Program Configuration menu to 'N'. However, keep in mind that if you disable automatic restoration, and then change the current logged drive and/or subdirectory while in an external program or DOS shell, Hypersignal will read (or attempt to read) a new configuration file, if any, after the external program finishes.

For real-time versions of the software, it may be necessary to access the configuration file to change the default memory and/or I/O base address for some types of DSP/Acquisition hardware. Use a text-editor to look at the "CONFIG.DOC" file (on distribution diskette #2) and find out which parameter in the configuration file to change. Also for real-time versions of the software, there exist a few other parameters affecting DSP/Acquisition board operation that cannot be controlled via data-entry menus, and may need to be accessed in the configuration file.

NOTE: One individual value for real-time software that will be mentioned here because of its use on several supported DSP boards is the "AIC control word", which is sent down to boards containing the TLC32044 AIC analog I/O chip at run-time. If you need to change aspects of this chip that can't be controlled from the data-entry menus, then you can alter this value directly in the configuration file. In the case of the "gain bits" in this value, if you change these bits from their default value, data-entry menu control over gain will be disabled. See section 4.7.2.5, Input Range and Gain, for more information.

3.8 Creating Hypersignal Waveform Files

There are essentially six ways to create a waveform file in Hypersignal format where none existed before:

1) Acquire an existing waveform file in some other format using the File Acquisition function (see section 4.1.1, Acquiring Waveforms From Disk Files).

2) Acquire data continuously to disk, in real-time, from a supported DSP/Acquisition board (real-time software only—see section 4.1.2, Acquiring Waveforms From DSP/Acquisition Hardware).

3) Use the Difference Equations function to generate a waveform based on input from unit-step, delta, trig, noise, and other functions, combined with recursive input, or "feedback" (see section 4.6.1, Waveform Generation with Difference Equations).

4) Write an external program that, after performing whatever processing for which it was intended, generates its output directly to Hypersignal waveform file format (see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT).

5) Capture a display trace from the Waveform Display/Edit function or Digital Oscilloscope or Spectrum Analyzer instrumentation functions (real-time software only—see section 4.7.1, Using Instrumentation Displays) and store to disk.

6) Execute a Hypersignal function, such as Filter Construction, that creates a waveform file as output resulting from its normal operation and processing.

3.8.1 Creating Complex Waveform Files

There are essentially four ways to create a complex waveform file in Hypersignal format where none existed before:

1) Acquire an existing complex waveform file in some other format using the File Acquisition function (see section 4.1.1, Acquiring Waveforms From Disk Files).

2) *After modifying the appropriate DSP Source Code Interface* (see section 4.7, Using DSP/Acquisition Hardware with Hypersignal), acquire "analytically sampled" complex data continuously to disk, in real-time, from a supported DSP/Acquisition board (real-time software only—see section 4.1.2, Acquiring Waveforms From DSP/Acquisition Hardware).

3) Use the "j" operator in the Difference Equations function to generate a complex waveform based on input from unit-step, delta, trig, noise, and other functions, combined with recursive input, or "feedback" (see section 4.6.1, Waveform Generation with Difference Equations).

4) Write an external program that, after performing whatever processing for which it was intended, generates complex output directly to Hypersignal waveform file format (see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT).

3.8.2 Creating Multichannel Waveform Files

There are essentially three ways to create a multichannel waveform file in Hypersignal format where none existed before:

1) Acquire an existing multichannel waveform file in some other format using the File Acquisition function (see section 4.1.1, Acquiring Waveforms From Disk Files).

2) Acquire data multichannel continuously to disk, in real-time, from a supported DSP/Acquisition board (real-time software only—see section 4.1.2, Acquiring Waveforms From DSP/Acquisition Hardware).

3) Write an external program that, after performing whatever processing for which it was intended, generates multichannel output directly to Hypersignal waveform file format (see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT).

3.8.3 Creating Frequency Domain Waveform Files

Only the FFT Generation function creates frequency domain waveform files directly in Hypersignal. The Difference Equations function could also be used, with careful attention to framesize and organization. If you need to create frequency domain files external to Hypersignal, say for example to specify real and imaginary waveform files upon which an Inverse FFT Generation can then be executed, please see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT. Although not difficult to generate frequency domain files via external program, you must pay attention to the "frame-based" organization of frequency domain files. One popular example of creating frequency domain files outside of Hypersignal:

Generating complex (".REA" and ".IMA") waveform files that represent the desired magnitude *and phase* response for a filter. An Inverse FFT Generation function can then produce the appropriate time domain impulse response for filtering other waveforms (via Convolution function).

3.9 Finding Waveforms

As you begin to build up a large amount of waveform files during the course of your work with Hypersignal, it will become more difficult to locate waveform files. There are three methods you can employ to aide in finding waveform files while using Hypersignal:

1) Pop-up directory display

2) Utilities menu function “Directory Display”

3) DOS shell

Pop-Up Waveform Directory Display

Pop-up directory display appears during data-entry editor menus as an automatic window display in the System Status area (upper left of main menu) when a specified filename cannot be found. Also, you can manually invoke pop-up directory display from inside any data-entry menu by using the ‘PGUP’ key.

Pop-up directory display includes waveform filename and extension, length, and date of creation. The search strategy for different waveform extensions and/or more specific waveform filename may be changed via consecutive use of the ‘PGUP’ key (see section 3.4.2 above, Pop-Up Directory Display, for complete information on waveform filename extensions and the search strategy).

You may also temporarily enter “∗” or “?” wildcard characters in the waveform filename field(s) to further direct a waveform search; for example a ‘test∗‘ entry would include all filenames starting with “test”, and having the appropriate extension. As another example, an entry of ‘x?test’ would include all filenames starting with “x”, ending with “test”, and having any other character in between.

NOTE: To re-initialize pop-up directory display after you have changed the filename in the data-entry menu, or added a wildcard character (‘∗‘, ‘?’), ***you must reposition the cursor at the beginning of the waveform filename field and ‘Enter’ the field*****—then press the ‘PgUp’ key.**

Directory Display Menu Function

The Directory Display menu function, listed under the Utilities column heading on the main menu, offers a DOS-like directory search capability. Unless specified explicitly, the drive and subdirectory are assumed to be the current values shown in the User Setup area. Wildcard characters “?” and “∗” are allowed; display is shown one page at a time. You can use the PgUp and PgDn keys to see the previous and next pages of directory display, respectively. The last page displayed also includes a “bytes free” display, showing the current number of free space (in bytes) for the current drive. Because of the capability to page backwards, this function is limited to about 100 total pages of display.

If you do not enter a colon (“:”) in the FILENAME field for the Directory Display menu selection then Hypersignal will assume the DRIVE field from the User Setup area. If you do not enter a backslash (“\”) in the FILENAME field then Hypersignal will assume the SUBDIRECTORY field in the User Setup area. Explicitly entering the ‘:’ and/or ‘\’ characters overrides the assumed User Setup values.

For convenient reference, below are listed the basic waveform filename extensions used by Hypersignal:

filename.TIM:	Time domain waveform data file
filename.FRQ:	Frequency domain magnitude file
filename.PHZ:	Frequency domain phase file
filename.PWR:	Frequency domain power spectra file
filename.REA:	Frequency domain complex real file
filename.IMA:	Frequency domain complex imag file

DOS Shell

Of course, for the most complete file directory service, including all command-line switches offered by DOS in conjunction with the “DIR” command, you can temporarily “shell out” of Hypersignal to the DOS command line (‘F10’ key)—but you can only do this from the main menu, and not from inside data-entry menus. Also, when you do this, remember two things:

1) The current DOS path (drive and subdirectory) will most likely be something else than where your waveforms are located. If you change the current DOS path while in the DOS shell, and the RESTORE DOS PATH field in the External Program Configuration menu is disabled, *be sure and change it back.* Otherwise, Hypersignal may read an unwanted configuration file from the

new DOS path upon returning. See section 3.6, Invoking External Programs and the DOS "Hot-Key", for detailed information on the subject of changing the DOS path during DOS shell execution.

2) Type 'Exit' to return to the Hypersignal main menu once you are done in the DOS shell.

3.10 Invoking Macros

Hypersignal macros can be invoked in three ways:

1) From the main menu.

2) Remotely, outside of Hypersignal, from another program or batch file.

3) From a previous macro; called "chaining".

Invoking a Macro from the Main Menu

The 'F3' key is used to configure and execute (if ready) a macro from the main menu. When the 'F3' key is pressed, the Macro Manager menu appears, which works like any other Hypersignal data-entry menu, even though it is not listed under one of the three main menu column headings. When the 'Enter' key or a mouse button is pressed while the cursor is on the last field (or the 'CRTL-ENTER' key sequence is pressed at any time) when this menu is displayed, the current macro specified in the MACRO FILENAME field is compiled (if needed) and then executed. Input parameters (if any) used by the macro are also specified in the Macro Manager menu, as is the compiler mode and pathname information needed to find the macro file. Pressing the 'ESC' key from the Macro Manager menu will not execute the currently specified macro. On-line Help is available as usual by pressing the 'F1' key.

Depending on the current macro compiler mode specified in the Macro Manager, the macro may or may not be compiled before it actually executes. The macro compiler can be in one of three possible modes: 1) always compile, 2) compile only if necessary ("make mode"), where the macro source code file is only compiled if its date stamp is later than that of the macro object code file, and 3) never compile (use existing macro object code). The software default is the make mode.

All status (and error messages) concerning macro compilation and execution are displayed in the System Status area (upper left of main menu).

Note that up to 6 (six) input parameters can be specified in the Macro Manager. These input parameters can be used to pass input and output filenames, data-entry menu parameters, and numeric values to a macro, allowing it to be used for different waveforms without modification. Also, the descriptive menu labels can be customized for these paramters for each macro—see the ".MCF" file, below.

For any given macro, Hypersignal keeps track of three separate filename extensions:

1) Source file extension, which defaults to ".MAC", but can be otherwise if specified explicitly in the Macro Manager menu MACRO FILENAME field.

2) Object file extension, which is always ".MXE".

3) Configuration file extension, which is always ".MCF". The configuration file contains all Macro Manager menu entries for the macro in ASCII text form, and is saved whenever the macro is executed. Since this file is in ASCII form, it can be edited in order to customize the input parameter field labels—see section 4.10.1.2, Parameter Passing From the Macro Manager Menu.

See chapter 7, THE HYPERSIGNAL MACRO LANGUAGE, for detailed information on all aspects of macros and the macro language.

NOTE: Hypersignal searches for all macro files on the path (drive and subdirectory) specified in the Macro Manager menu. The User Setup path, or any other Hypersignal path, is not considered.

Invoking a Macro Remotely (Outside of Hypersignal)

To invoke a macro from outside of Hypersignal, the macro source filename is passed as a command line parameter in the HSxxxxx.EXE file specification. Hypersignal first ensures that the macro source file exists, and, if so, looks for a macro config-

uration file (see below) with which to initialize the Macro Manager menu, and then starts the macro. See section 3.0.1, Advanced Features When Starting Hypersignal, for examples of command line parameter specifications. Note that the macro language contains "Abort" and "Exit" commands that allow the program to return from Hypersignal back to the caller, completing a "remote execution" of a macro.

Invoking a Macro from Another Macro

The Macro Manager menu can be invoked from inside an executing macro by a "Macro Manager" statement, followed by input parameters (if any), interactive commands (if any), and a standard "endfunc" statement (see section 7.3, Invoking Hypersignal Menu Functions From Macros). Re-invoking the Macro Manager menu from within an executing macro is an advanced technique that can be used for either of two purposes, or both:

1) Allow the input parameters to the current macro to be changed dynamically. To do this, a $ESC interactive command should be entered prior to the "endfunc" statement, so that the macro does not run "recursively". Changing input parameters dynamically is not recommended as standard macro programming procedure, but could be useful in some situations.

2) To chain another macro to the end of the currently executing macro. To do this, a "Macro Manager" statement, followed by the appropriate new macro filename, any input paramaters needed, and an "endfunc" statement, should be entered in the macro as the *last* executable macro command.

this page intentionally left blank

4.0 USING HYPERSIGNAL FUNCTIONS

This chapter is intended to provide you with the bulk of the knowledge required for accomplishing specific tasks in Hypersignal. The chapter has been organized on a "how-to" basis, covering all DSP processing and display that can be achieved in Hypersignal. Because of this organization, some sections may cover more than one actual Hypersignal menu item, as they explain from start to finish how to perform a certain type of processing or display. Thus, individual menu items are not necessarily listed one-by-one in this chapter. For a simple, item-by-item reference to specific menu selections and their data-entry fields and interactive display commands, see the "Hypersignal-Macro Software Reference Guide".

4.1 Getting Waveforms In and Out of Hypersignal

What good is a DSP program if you can't get data into it? There are several ways to get waveform data into Hypersignal:

1) File Acquisition: acquiring waveforms from disk files generated by some other source or program

2) A/D Conversion: acquiring waveforms directly from DSP/Acquisition hardware (real-time software only)

3) ASCII Import: importing ASCII waveform data that was previously exported by Hypersignal or that has already been organized in the Hypersignal waveform format

For a complete list of ways to actually generate different types of Hypersignal waveforms from scratch, see section 3.8, Creating Hypersignal Waveform Files.

Of course, you can also move waveforms out of Hypersignal:

1) D/A Conversion: sending waveforms directly to DSP/Acquisition hardware (real-time software only)

2) ASCII Export: exporting Hypersignal waveforms to external ASCII format files, for use with other programs or text-editors

4.1.1 Acquiring Waveforms From Disk Files

The File Acquisition function provides the means to import waveform files with different organizational and input data formats to standard Hypersignal waveform files ready for subsequent signal processing analysis. Input waveform data may also come from a serial channel or direct PC memory-access.

The File Acquisition function is very flexible, allowing parameters such as data format, sample width (specified in nibbles, or 4-bit increments), number of samples to discard, and mean subtraction, to be specified. Also available is the ability to strip out samples from an interleaved waveform file, mask samples left or right (maintaining sign value) to remove embedded control or channel information, acquire multichannel or complex data, and acquire μ-Law or A-Law data.

The File Acquisition function can also import waveform data directly from files created by various DSP chip manufacturer development utilities, depending on the current "Development DSP" selection in the "System Configuration" menu—see section 2.5.2, Initializing the System Configuration Menu.

The inherent difference between the File Acquisition function and the ASCII Export/Import function is that File Acquisition *creates a Hypersignal file from scratch—both waveform data and header*. The ASCII Import function requires the ASCII input file to be *already organized in Hypersignal format and to already include a header*, whether it was created by the ASCII Export function, spreadsheet program report generator, text-editor, etc. See chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT, for more information about waveform headers and data values.

NOTE: **Hypersignal-Macro software can represent waveform file data with magnitude ranging from 2.93873e-39 to 5.575e+42. If the maximum absolute value of the data is less than 100 or greater than 32767, then the File Acquisition function will prompt you to perform "auto-ranging", in which the function re-acquires the data, scaling it up or down. Any such scaling is then represented using a non-zero "waveform amplitude exponent" in the waveform header. This storage approach allows both fast throughput, especially with real-time hardware functions, and also fairly good dynamic range. See chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT, for more information about waveform data values and waveform header exponents.**

File Acquisition Menu Fields

The following sections explain the File Acquisition menu data-entry fields:

INPUT FILENAME:

This field should specify the name of the input file containing sampled data to be imported to Hypersignal binary format. The filename must be a valid DOS name (up to 8 characters) with a valid DOS filename extension (period followed by up to 3 chars). Only if the filename truly has no extension should it be omitted. Also, unlike all other data-entry menu filename fields in Hypersignal, a disk drive identifier can be entered in front of the filename; an example entry for such a case might be 'A:example.dat'. This is done in the File Acquisition function alone to facilitate import of waveform files from diskette, which are often used to send waveform data between groups working with different systems and/or different signal processing programs.

The filename you enter in the INPUT FILENAME field is also used to store a "setup save" file, which stores all data-entry fields in the File Acquisition menu for future reference. Then, if you need to re-import, using the same menu parameters (or slightly changed ones), all previous settings will be recalled when you "load" that waveform filename. File Acquisition setup save files are stored with ".ACQ" filename extensions, in ASCII text format.

Alternatively, you can enter 'SP' as a filename if the input data source is to be a serial channel. See the "CONFIG.DOC" file for information on choosing and configuring serial characters via the "HSMACRO.CNF" configuration file. For serial channel acquisition, begin sending serial data *only after* display of the ready message "ACQUIRING FRAME — OUT OF N".

Finally, you can enter 'SSSS:OOOO' as a filename, where "SSSS" and "OOOO" are 4-character hexadecimal segment and offset values, respectively, if the source of input data is to be PC memory.

OUTPUT TIME-DOMAIN FILENAME:

Enter the name the resulting Hypersignal file should be stored under. A ".TIM" extension will automatically be appended to the output filename—an error message will be displayed if any extension is entered explicitly.

FORMAT:

Controls the format that will be used when translating input data to Hypersignal binary data. Select one of the following possible entries:

'DSP' Entering 'DSP' enables acquisition from DSP chip development system files implied by the "Development DSP" choice in the "System Configuration" menu, as enumerated below:

A 'DSP' entry will acquire Texas Instruments 320-series EVM / XDS / SWDS (see section 4.1.1.1 below) software simulator and hardware emulator format output files when the "Development DSP" field is set to 'TI10', 'TI20', 'TI25', or 'TI30'.

A 'DSP' entry will acquire AT&T WE DSP32-SL software simulator and DSP3EMU emulator output format files (see discussion on AT&T tools in section 4.1.1.1) when the "Development DSP" field is set to 'WE32' or 'WE32C'.

A 'DSP' entry will import Motorola SIM56000 software simulator or DSP56000 ADS hardware emulator files when the "Development DSP" field is set to 'M56' (see discussion on Motorola tools in section 4.1.1.1).

A 'DSP' entry acquires Analog Devices ADSP-210x software simulator (DSPPS) output files when the "Development DSP" field is set to 'AD21' (see notes on Analog Devices in section 4.1.1.1).

A 'DSP' entry acquires NEC DDK77220 board emulator output files when the "Development DSP" field is set to 'N20'.

'BIN' A format entry of 'BIN' will import general binary fixed-point data files. Use the SAMPLE WIDTH field, below, to specify how many bits are in each input sample.

'SWP' Also acquires binary data files, but with byte halves of word values stored in a "swapped" manner compared to Intel convention—a format used by the Motorola 680x0 and other popular microprocessors.

'DEC' Acquires ASCII decimal "0" through "9" and "-" characters. Some examples of valid ASCII decimal format data samples:

-105
32767
0
+479
6983
-1
-29754

'HEX' Acquires ASCII hexadecimal "0" through "9" and "A" through "F" characters. Some examples of valid ASCII hexadecimal format data samples (4 nibbles):

FFFF
0
abc
$1234
F000H
BA34H
77

Some examples of valid ASCII hexadecimal format data samples (6 nibbles):

123456
999
0
fedcba
123456H

'FP' Acquires ASCII floating-point "0" through "9" and "E", ".", "+", and "-" characters. Some examples of valid ASCII floating-point format data samples:

0.17892
3.45678E-09
0.8928E+12
5.67E-10
0.0
10
1e5
-.00009876
1.2345e07

NOTE: **Two special "format attributes" may be attached to the FORMAT field entry. For all formats, entering a 'U' after the format selection (e.g. 'BINu' or 'DECU'), will cause the program to treat the incoming data as unsigned. When a 'U' is entered, the value of SAMPLE WIDTH, below, becomes important in the conversion to signed data.**

For binary formats with SAMPLE WIDTH ≤2 only, entering an 'L' or 'A' after the format will cause μ-Law or A-Law expansion, respectively. Examples: 'BINL', 'BINa', etc.

SAMPLE WIDTH:

Entry in nibbles (nibble=4 bits, or 1/2 byte) specifies sample width for binary, swapped binary, and ASCII hexadecimal formats (see FORMAT field above).

For binary formats, SAMPLE WIDTH specifies literally the number of nibbles per sample from the incoming byte stream. For ASCII hexadecimal format, SAMPLE WIDTH determines *the maximum 2's complement positive and negative values that will be acquired*, but places no limits on the character width of each sample. For example, SAMPLE WIDTH=4 would cause values

above 32767 (7FFFH) to be considered negative. *For ASCII decimal and floating-point formats,* SAMPLE WIDTH *is ignored,* as the translation algorithm depends on separator characters to determine the end of a sample.

NOTE: **SAMPLE WIDTH *is not ignored for ASCII decimal and floating-point formats in one case: the 'U' (unsigned data) option has been attached.* In all unsigned data format cases, the value of SAMPLE WIDTH is used to calculate the value to be subtracted from the incoming data as follows:**

sample = sample - $2^{(sample_width*4 - 1)}$

INITIAL BYTES/SAMPLES TO DISCARD:

Sets the number of initial bytes or samples to discard at the start of the input data. *In the case of binary or swapped-binary format (see* FORMAT *field above) this value should be specified in bytes, regardless of sample width.* Note this provides the capability, however unlikely, to construct word samples from odd boundaries. In the case of all other formats (i.e. ASCII formats), this value should be specified in samples.

NUMBER OF INPUT SAMPLES:

Setting this field to any value other than zero specifies explicitly the number of samples to acquire. Setting this value to zero (0) causes the program to input all data from the input file. Cases where an explicit value is needed include serial channel acquisition and files from which not all data is desired. The maximum number of samples that will be acquired via serial port is 16384; for files as input, the maximum number is 536,870,912, allowing binary file lengths over 1 Gigabyte (for ASCII input files, sample-length may be difficult to correlate with file-length, because of the varying number of characters per sample).

SAMPLING FREQUENCY:

Specifies the sampling frequency (in Hertz) for the output time domain file. Sampling frequency can be a maximum of 1,073,709,056 Hz (1.07 GHz) and is rounded to the nearest Hertz when stored in the acquired file's waveform header.

SUBTRACT MEAN:

'Y' enables mean (average) subtraction for the input data, and 'N' disables mean subtraction. The mean is calculated as a 16384-sample running sum over all input samples. For acquisition lengths less than 16384, the mean subtracted will be an exact value. For lengths greater than 16384, the running sum will provide an extremely good approximation to exact mean subtraction, since the current estimated mean value will change very slowly.

CHANNEL STRIP,MASK:

Entering one or both of these parameters to be non-zero allows waveform data to be acquired from a file by stripping every Nth sample from the file, and/or masking off bits at either end of each acquired sample. Entry should be in the form

S,M

where "S" is the STRIP value and "M" is the MASK value.

S=0 indicates no stripping, S=1 directs every other sample to be used, S=5 indicates every sixth sample, etc. Non-zero stripping values can be especially useful in the case of time-division-multiplexed data files, where data has been interleaved (a common example: multichannel A/D hardware).

A positive value of M will cause M bits at the high end of the sample (M msbs) to be removed and the sample sign-extended. A negative value of M will cause M bits at the low end of the sample (M lsbs) to be removed and sign-extension to be maintained. Note that the effect of negative values of M is essentially a scaling operation, or divide by 2^M. This type of operation may also useful in the case of multiple-channel acquisition, where source channel information has been stored as part of each sample. The maximum number of sample bits that can be masked is 12. If M=0, no masking will be performed.

FRAMESIZE,FFT ORDER,NUM. CHANNELS,COMPLEX:

NOTE: The following four (4) values can be termed "POST-ACQUISITION"; they have no effect on the actual file acquisition process itself, but are stored in the waveform header and subsequently referred to by Time and Frequency Domain processing functions. The FRAMESIZE and FFT ORDER values are also used by the program to automatically initialize some functions, such as Waveform Display/Edit and FFT Generation. Thus, to ensure proper post-processing of your acquired data, and for your convenience in allowing the program to initialize some data fields upon loading the waveform, you can enter these parameters at the time of waveform acquisition.

The FRAMESIZE value specifies the analysis framesize (in samples) for the acquired Hypersignal time domain file. FRAMESIZE is used to initialize the Waveform Display/Edit, FFT Generation, and LPC Autocorrelation functions. Minimum entry is 2 and maximum is 32767, although the Waveform Display/Edit function has no maximum limit on framesize entry. A zero entry will keep the program from auto-initializing these functions.

The FFT ORDER value specifies the default FFT size (FFT size is $2^{\text{FFT ORDER}}$) to be used with subsequent FFT Generation operations. Minimum entry is 2 and maximum is 12. If you are not sure of the FFT size that will be required in your processing, enter '0' (zero)—this will keep the program from initializing future FFT ORDER fields to a default value.

The NUMBER OF CHANNELS value specifies the number of distinct time-division-multiplexed (interleaved) channels inside acquired waveform. Default entry is '1' (single-channel data). Hypersignal functions treat two-channel waveforms automatically, displaying and processing the two channels separately. For example, by referring only to the name of the two-channel ("stereo") waveform you acquired, the Waveform Display/Edit function will automatically show dual-channel traces, the FFT Generation function will automatically process each channel separately, Convolution and Recursive Filtering will automatically filter each channel separately, and frequency domain displays will automatically display dual-channel traces. For more than two-channel waveform files, you may want to "strip" the input waveform into separate output waveform files (see the CHANNEL STRIP,MASK field, above). Minimum NUMBER OF CHANNELS entry is 1 and maximum entry is 64.

A COMPLEX entry of 'Y' or 'C' will cause the complex attribute to be set in acquired waveform file header. Hypersignal expects complex data as real, imag, real, etc. For multichannel complex data, Hypersignal expects real0, imag0, real1, imag1, real0, imag0, etc. where 0 and 1 indicate channels 0 and 1. Entry of 'N' or 'R' specifies real, or noncomplex data. Default entry is 'N'.

4.1.1.1 Acquiring DSP-Chip Manufacturer Development Tool Files

Below is somewhat more detailed information on both DSP development tool file acquisition and direct memory-mapped DSP board data acquisition, if applicable.

AT&T Development Tool File Acquisition

When acquiring AT&T DSPxx data, keep the following guidelines in mind: the 'DSP' selection is designed to work with *both* DSP32-SL software simulator *and* DSP3EMU emulator output when the "Development DSP" field in the "System Configuration" menu is set to 'WE32'. In this case, 'DSP' causes the program to first look for ASCII floating-point data in DSP3SIM (software simulator) format; if data does not match as expected the program re-attempts acquisition assuming DSP3EMU (emulator) binary floating-point format. Non-floating-point data written to disk from DSP3SIM requires selection of an appropriate general-purpose format *other* than 'WE' (e.g. 'BIN' for binary integer). For information on direct, real-time data acquisition from DSP32x based hardware, see section 4.1.2.

Texas Instruments Development Tool File Acquisition

When acquiring files created by Texas Instruments 320xx development tools, the 'DSP' selection is designed to automatically detect the format in use and adjust accordingly. The file formats for each 320xx generation of development tools vary greatly, and, in some cases, even vary within a generation; for example, between the software simulator and hardware emulator, and different output file format options in one development tool. In any case, it is highly recommended that standard, non-'DSP', format selections, such as 'HEX' and 'DEC', be avoided, as quirks in the various 320xx formats will not allow the standard formats to work reliably.

Texas Instruments SWDS Board File Acquisition

When accessing the Texas Instruments SWDS software development board, it is necessary to halt the 320C25 processor while within the SWDS environment. After exiting the SWDS environment (with SWDS board halted) and invoking Hypersignal, data located in the SWDS external program memory or data memory spaces may be imported via the File Acquisition function into the Hypersignal environment using direct-PC memory (SEG:OFS) acquisition. Usually, SWDS dual-port memory is located at segment E (E000:0000H, or SEG=E000H, OFS=0) for PC/XT machines and segment D for AT machines (jumpers P1-P4 on the board set the segment address). The 24 kWords of SWDS memory is mapped into program and data space in four possible combinations, and can thus be accessed by varying the offset (OFS). Below is a table delineating SWDS memory boundaries assuming segment E:

prog memory size	SEG:OFS	data memory size	SEG:OFS
8 kWords	E000:2000	16 kWords	E000:6000
12 kWords	E000:2000	12 kWords	E000:8000
16 kWords	E000:2000	8 kWords	E000:A000

Note that program memory always starts at 2000H. This is inherent in the SWDS board. Remember, 80x86 and 80x88 machines are byte-oriented, while the 32020 and 320C25 are word-oriented machines, so use caution in calculating offset values (PC address = 2∗TI320 address). For example, if one was interested in acquiring a data buffer from SWDS external data memory which started at the beginning of page 12, and the memory map was set for 12 kWords of program memory and data memory each, then the value entered for SEG:OFS in Hypersignal File Acquisition would be calculated as follows:

SEG = E000H
OFS = 8000H + page 12 ∗ 128 ∗ 2 = 8C00H

Thus the INPUT FILENAME field would be entered as ‘E000:8C00’.

Motorola Development Tool File Acquisition

When acquiring Motorola DSP5600x development tool output files, keep the following guidelines in mind: the ‘DSP’ selection is designed to work with *all* SIM56000 software simulator output formats (ASCII floating-point, hexadecimal and decimal). *and also* DSP5600x ADS emulator output, when the “Development DSP” field in the “System Configuration” menu is set to ‘M56’. The program first looks for ASCII floating-point data, then ASCII decimal, then ASCII hexadecimal. Note that the standard ‘FP’ format selection could also be used to acquire floating point data, as could ‘HEX’ and a SAMPLE WIDTH entry of 6 to acquire hexadecimal data; however, the ‘DSP’ selection will be most convenient. For information on direct, real-time data acquisition from DSP5600x-based hardware, see section 4.1.2.

Analog Devices Development Tool File Acquisition

When acquiring Analog Devices ADSP-210x development tool output files, keep the following guidelines in mind: the ‘DSP’ selection is designed to work with *all* DSP software simulator output formats (ASCII hexadecimal, binary) when the “Development DSP” field in the “System Configuration” menu is set to ‘AD21’. Note that the standard ‘HEX’ format selection and a sample width entry of 4 could also be used to acquire hexadecimal data; however, the ‘DSP’ selection will be most convenient. For information on direct, real-time data acquisition from ADSP-210x and ADSP-21020 hardware, see section 4.1.2.

4.1.2 Acquiring Waveforms From DSP/Acquisition Hardware

(Real-Time Software Only)

The Analog Conversion menu selection enables conversion of analog to digital data (A/D) and digital to analog data (D/A) in real-time (continuously, with no loss of data) to and from disk. The DSP/Acquisition hardware that will be used should be specified in the System Configuration menu (see section 2.5.2).

The maximum rate of data transfer depends on the type of disk (hard disk or RAM disk), sampling rate, and number of simultaneous analog channels.

Acquired waveform files are then available for later analysis, DSP processing, display, and eventual reconversion back to analog form. Since transfer is continuous, waveform file length is limited only by the capacity of the storage media in use. Analog conversion parameters such as channel selection, real-time filters, signal gain/attenuation, multicondition trigger type and threshold, sampling frequency, time domain analysis framesize, and signal offsets are under user control.

Before describing how to use the Analog Conversion function, please be aware that basic discussion about choosing and using different types DSP/Acquisition hardware is presented in the Hardware Reference Guide. It is very important to have a fundamental understanding of the underlying hardware that will be performing the data conversion and transfer. The following topics as they relate to different types of hardware are covered in the Hardware Reference Guide:

1) Sampling Rate/Data Transfer Considerations

2) Data Resolution

3) Anti-Aliasing

4) Number of Channels

Several "subfunctions" are active inside the Analog Conversion function. These include signal generation (D/A output), stimulus & response, and real-time filtering. While mentioned here, these functions are discussed in greater detail in their own sections (4.1.3, 4.7.4, 4.9.3, respectively).

Analog Conversion Menu Fields

The following sections explain the Analog Conversion menu data-entry fields:

TIME DOMAIN FILENAME:

Name of time domain waveform data file to either write incoming analog-to-digital samples to, or read outgoing digital-to-analog samples from.

This field specifies a standard Hypersignal format time domain waveform file. Use the 'PgUp' key to view pop-up directory of current time domain files. Use the 'PgDn' key to scroll through current pop-up directory display. Consecutive use of 'PgUp' key will change the search strategey for current pop-up directory display (see section 3.4.2, Pop-Up Directory Display).

The filename you enter in the TIME DOMAIN FILENAME field is also used to store a "setup save" file, which stores all data-entry fields in the Analog Conversion menu for future reference. Then, if you need to re-acquire or re-generate, using the same menu parameters (or slightly changed ones), all previous settings will be recalled when you "load" that waveform filename. Analog Conversion setup save files are stored with ".ADC" filename extensions, in ASCII text format.

ACQUIRE/GENERATE:

An 'A' entry indicates signal acquisition (analog-to-digital conversion); a 'G' indicates signal generation (digital-to-analog conversion). Also, you can enter 'AM' to cause the digital input to be "looped back" to the digital output (if supported by the current hardware) before and during acquisition. This allows you to "monitor" input, deciding where to be begin acquisition (by answering the "BEGIN ACQUISITION ?" prompt). Loopback is always digital with a one-sample delay; that is, samples read in by the DSP chip are written back out one sample period later. A 'GM' entry is the same as a 'G' entry.

After a 'G' or 'GM' entry, you can also enter an 'L', which will cause the digital-to-analog conversion output process to repeat when finished. The process will repeat until you press the 'Esc' key. Entering an 'L' after an 'A' or 'AM' entry has no effect.

After an 'A' or 'AM' entry, you can also enter a "stimulus" waveform filename. If you do so, then the software will download the stimulus waveform to memory on the DSP board in use, and output this waveform at the same time as it is acquiring the input waveform. See the TRIGGER field below for use of negative or positive acquire delays with respect to the onset of the stimulus. If digital loopback is enabled ('AM' entry), then loopback will occur until the "BEGIN ACQUISITION ?" prompt is answered. A stimulus waveform entered after a 'G' or 'GM' entry has no effect. See section 4.7.4, Using "Stimulus & Response", for more detail.

NOTE: **If you attempt to overwrite a previously existing waveform file when acquiring data, a very noticeable audible warning will sound. You can disable this warning by turning off the global SOUND field in the User Setup menu selection. However, you should remain as careful as possible when overwriting existing waveform files, in order to avoid destroying a valuable one.**

CHANNEL LIST:

The CHANNEL LIST field specifies the analog channel list for input or output data. For hardware that supports multiple channels, enter desired channels as 0 to N-1 (for example, '0', or '0,1'), where N is the number of hardware channels, separated by a ',' or '/', or other nonnumeric character. Use the '-' character, as in '0,1-4,7', to specify a descending or ascending range. For analog hardware that supports only one channel, entering other channels may produce undefined results.

Also, in addition to the channel list entry, you can enter the name of an FIR or IIR filter to be "snapped-in" for filtering incoming or outgoing data in real-time. Simply enter the filter filename after the channel list, for example '0,1,elliptic'. Also, for fixed-point DSPs, a quantization can be specified, for example '0,1,elliptic,14'. Normally, snap-in real-time filters will have been previously designed using the Hypersignal Filter Construction functions; however, filter coefficient specification files can be modified directly in order to apply virtually any filter in real-time (see section 4.8.3, Specifying Filters Outside of Hypersignal). Note that you cannot apply different filters to different analog input channels; however, you can do this in the Spectrum Analyzer and Digital Oscilloscope functions. Also note that the snap-in filter function currently works only for one and two-channel operation—additional channels would require modification of the appropriate "DSP Source Code Interface" (see the Hardware Reference Guide). Please see section 4.9.3, Using "Snap-In" Real-Time Filtering, for much greater detail on using real-time filters with Hypersignal.

NOTE: **For hardware that is upgraded or modified from its standard configuration to include additional analog channels, modification of the appropriate "DSP Source Code Interface" can be performed to activate the new channels. See the Hardware Reference Guide for details.**

Do not increase SAMPLING FREQUENCY, ***below, to compensate for multichannel operation.*** If any programming of the hardware currently in use with increased sampling frequency is required for multichannel operation, it will be performed automatically by the software.

When acquiring a multichannel waveform data file, Hypersignal always stores the data in interleaved form, starting with the first channel entered in the CHANNEL LIST. For example, a CHANNEL LIST entry of '0,1,2' would cause waveform file data to be stored as chan0, chan1, chan2, chan0, chan1, chan2, etc, repeating in groups of three.

All Hypersignal processing and display functions can automatically handle two-channel waveforms. For example, by referring only to the name of the two-channel ("stereo") waveform you acquired, the Waveform Display/Edit function will automatically show dual-channel traces, the FFT Generation function will automatically process each channel separately, Convolution and Recursive Filtering will automatically filter each channel separately, and frequency domain displays will automatically display dual-channel traces. After acquiring a two-channel or more waveform file, the Analog Conversion function will prompt you to perform a "stripping" operation. If you answer the prompt positively, Hypersignal will generate, without destroying the original interleaved waveform, a single-channel waveform file for each channel entered in the CHANNEL LIST field. The stripped waveform filenames are created by adding the channel number to your entry in the TIME DOMAIN FILENAME field, above. For example, if you entered "test" as your filename, then stripping with a CHANNEL LIST entry of '0,1,2' will cause "test0", "test1", and "test2" single-channel waveform files to be created. Thus, for three-channel or more waveform files, you may want to answer the stripping prompt positively, and generate separate channel waveform files.

Please note that the channel list order is not affected by TRIGGER CHANNEL LIST, below, regardless of whether the lists are in different order, or even if an entry on the TRIGGER CHANNEL LIST is not one of the actual channels being acquired (being used for triggering only).

INPUT,OUTPUT GAIN/ATTENUATION:

This field allows you to control gain (or attenuation) for both input and output data. In most cases where DSP-based hardware is being used, gain or attenuation is achieved strictly through software scaling (implemented by the DSP chip). However, in a few cases with DSP-based hardware, and also in some cases with acquisition hardware not using a DSP chip, actual hardware-implemented gain values can be specified. Of course, if you are using acquisition hardware with no DSP chip, then this

field will be inactive if the hardware does not contain programmable hardware gain. Currently, however, programmable hardware gain is still the exception and not the rule—see the Hardware Reference Guide for the final word on how this field should be used for specific DSP/Acquisition hardware.

For standard DSP software-controlled scaling, entry should be in the form of 'inp,out', where "inp" is the signal input gain or attenuation value, and "out" the output value. Maximum signal gain is 256 and maximum attenuation is 1/256 (or approximately 0.00391).

NOTE: When performing signal generation with hardware containing a DSP chip, the software will automatically multiply the entered output scaling value by another scaling value equal to $2^{\text{AMPLITUDE EXPONENT}}$, where AMPLITUDE EXPONENT is contained in the output waveform header. Only if the amplitude exponent is non-zero will the entered value be affected. See chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT, for more information about waveform amplitude and exponent values.

In some cases, the actual label for this field may include the word "LIST", in which case you can enter a "gain list" that applies to the channels entered in the CHANNEL LIST field. For example, one unit might have programmable input gain in dB for its two channels; in this case, an entry of '20,40,1' would specify 20 dB of gain for the first channel in the CHANNEL LIST, 40 dB for the second, and a *linear* software scaling factor of 1 for output data. As another example, another unit might allow completely independent gain values for 8 different channels—you might then enter '1,2,4', which would specify a gain of 1 on the first channel in the CHANNEL LIST field, a gain of 2 on the second, and a gain of 4 on the remaining channels.

TRIGGER CHANNEL LIST:

Entry in this field is similar to CHANNEL LIST, above, except that values indicated here specify the trigger sources and sequence necessary to begin acquisition.

Trigger conditions (level, delay if applicable) must be met for data coming in on the entered channels before *any* acquisition begins, regardless of whether this channel is represented in the CHANNEL LIST field or not. Obviously, for hardware supporting only one analog input channel, then CHANNEL LIST and TRIGGER CHANNEL LIST should each be one entry, and also be the same. However, for hardware supporting more than one channel, it may be possible to specify that trigger must occur on *any or all* TRIGGER CHANNEL LIST channels specified, and also in the sequence specified. Examples such as triggering on one channel and acquiring on another, triggering and acquiring on one channel and acquiring on others, and triggering on more than one channel may be possible. Normally in such a case, there is a hardware limit on the maximum number of trigger channels (or "nested conditions") that may be put into effect at one time—a limit of two (2) is typical. Consult the Hardware Reference Guide to see what the possibilities are for specific DSP/Acquisition hardware.

Usually, the number of entries in the TRIGGER CHANNEL LIST field cannot be more than there are channels, but there may be exceptions: some hardware support "trigger channel numbers" greater than the highest data input channel—these may refer to special triggers, such as the AC power line, special analog triggers, etc. Consult the Hardware Reference Guide for information in this area.

The TRIGGER CHANNEL LIST field is not active during signal generation (D/A output) operations.

TRIGGER MODE (EXT/INT, LEVEL, DELAY):

This field is highly hardware-dependent. The only part of this field that is consistent for a broad range of hardware is the LEVEL entry, which controls the software threshold used for DSP-based hardware, and the DELAY entry, which controls the acquisition delay when stimulus output is used.

Whether trigger should be internal or external is dependent on the hardware hardware currently in use. If the current hardware does not support an external trigger, then an 'EXT' entry will have no effect. Internal triggering means that the board currently in use, either through DSP software processing in real-time, voltage-comparators implemented in hardware, or through direct software control, will determine trigger based on input signal data to the board through its A/D channels. External triggering means that the activation signal or level will come from outside the "channels", on some other connector (like a TTL-level input). See the Hardware Reference Guide to determine which hardware actually possess external triggering capability (most hardware doesn't).

NOTE: For DSP-based hardware, a TRIGGER MODE entry of 'INT,0,0' essentially turns triggering off, independent of current TRIGGER CHANNEL LIST values. An entry of 'EXT,0' would turn triggering off only if the hardware has an external trigger *and* it can be programmed with a level; otherwise external triggering would still be enabled.

Whether trigger level is implemented in hardware on in DSP software is dependent on the hardware hardware currently in use. In most cases where the hardware is DSP-based, trigger level entry should be made directly in A/D sample levels (e.g. -4095 to 4095 for a 12-bit A/D, or -32767 to 32767 for a 16-bit A/D). In a few DSP-based cases, and with acquisition-only hardware that supports programmable hardware triggering, entry may be possible in units of Volts, using floating-point format. Check the Hardware Reference Guide to see if this is the case.

With DSP-based hardware, a positive LEVEL entry will cause trigger to occur when the first input data sample is read greater or equal to that value; a negative LEVEL entry will cause trigger to occur when input is less than or equal. In cases where trigger level is implemented in hardware, then actual "rising edge" or "falling edge" type operation may be possible.

Whether an actual trigger DELAY (in seconds) can be specified is dependent on the hardware hardware currently in use. In a few cases, an actual trigger delay may be specified that controls the onset of signal acquisition (some time either before or after the trigger occurs). Again, consult the Hardware Reference Guide. In the case of all DSP-based hardware, the DELAY portion of the TRIGGER MODE field is used to specify a stimulus delay, in units of seconds. A positive delay will cause acquisition to start at DELAY seconds *after* the stimulus has begun; a negative delay will cause acquisition to start delay seconds *before* the stimulus begins.

NOTE: For stimulus & repsonse operations, all other triggering parameters (level, channel, etc.) take effect *after* the stimulus delay expires (when acquisition begins).

The TRIGGER MODE field is ignored during signal generation (D/A output) operations.

NUMBER OF SAMPLES:

This field specifies the number of time waveform file data samples to store when acquiring input data, and the number of time waveform file samples to read when generating output data.

When acquiring input data, the actual length of the input file will be the NUMBER OF SAMPLES specified times the number of channels entered in the CHANNEL LIST field.

When generating signals, the program will query you if the NUMBER OF SAMPLES entry does not agree with the NUMBER OF DATA VALUES parameter in the waveform file header (see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT). As with input, NUMBER OF SAMPLES is first multiplied by the number of channels entered in the CHANNEL LIST field. Finally, if the actual waveform file length turns out to be less than the final NUMBER OF SAMPLES value, signal generation stops after the actual waveform length has been exceeded.

To figure the total amount of time for which data will be acquired/generated, multiply NUMBER OF SAMPLES by SAMPLING FREQUENCY, below.

SAMPLING FREQUENCY:

This field specifies the sampling frequency, in Hertz, for signal acquisition or signal generation operation. Depending on the hardware and the scheme used to generate various sampling rate clock frequencies, the actual sampling frequency may differ somewhat from the entered value. Usually this is due to the use of integer frequency dividers, or a log-companding (or similar) scheme to achieve a wide dynamic range of possible sampling frequencies. During the signal acquisition/generation process, the sampling frequency *actually being used* is displayed onscreen.

NOTE: Do not increase SAMPLING FREQUENCY to compensate for multichannel operation. If any programming of the hardware currently in use with increased sampling frequency is required for multichannel operation, it will be performed automatically by the software.

If the requested sampling frequency exceeds the rate specified by the hardware manufacturer, or the overall transfer rate to disk of your system, *you may or may not be warned*. For some hardware, the Hypersignal device drivers can detect accurately when an overrun condition has occurred, and display an error message. For most hardware, however, including most DSP-based hardware, discontinuous data (data with gaps) must be detected using FFT Generation and the 2-D Spectrograph display. By applying an FFT with 50% overlap and a Hamming or Hanning window, any data discontinuities will appear as "wideband noise", or as "vertical stripes" in a 2-D Spectrograph display of the entire waveform.

Because systems can vary so dramatically, it is difficult to predict exactly the maximum sampling frequencies that will be possible. Besides pure machine and disk access speed, the state of waveform file storage on disk may affect data transfer rate as well. If file storage is highly fragmented, causing the disk drive to perform long seeks between sections of the waveform, maximum sampling frequency may be reduced. Note that in Hypersignal, if a waveform file will require more than its existing file space, the space is created ahead of time to pre-process any seeks to the disk directory or any other area to locate new file space. In general, the more one can guarantee that input waveform files are contiguous, the more one can guarantee the maximum sampling rate, especially for very large waveform files. Every so you often you should run a "disk pack", or "optimizer" program to free up large sections of contiguous space. You should note if the optimizer utility lists any "bad tracks" or "bad blocks" within contiguous sections of free space. A section contiguous in terms of files may in reality become non-contiguous, because when bad tracks are encountered in actual use, the disk controller will seek to alternate tracks which are usually located at the physical center of the disk, causing long seek delays and lowering of overall sampling rates.

NOTE: Hypersignal internally turns the DOS "VERIFY" setting to OFF during signal acquisition operations, and restores it to its previous state when finished. This is done because a VERIFY state of ON causes DOS to verify every write to disk with a read, reducing the overall rate at which data can be written to disk.

When generating signals, the program will query you if the SAMPLING FREQUENCY field entry does not agree with the "SAMPLING FREQUENCY" parameter in the waveform file header (see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT).

Buffer sizes used by analog functions are reduced to achieve an approximate transfer rate of 1 buffer/sec when the sampling rate is less than the maximum buffer size allowed by the DSP/Acquisition hardware. This provides greater resolution in acquiring the requested amount of data when sampling rates are very low, and eliminates time that would be spent waiting for huge buffers to fill with extra, unneeded data in very-low-rate operations.

FRAMESIZE:

The analysis framesize (in samples) that will be used to initialize subsequent Waveform Display/Edit, FFT Generation, and LPC Autocorrelation time domain operations. *This field has no effect on the actual acquisition or generation process,* making entry here more a matter of convenience than anything else. Enter a '0' if you wish to keep the software from initializing time domain functions with a framesize value for acquired waveforms. The maximum FRAMESIZE entry is 32767.

INPUT,OUTPUT OFFSET:

Values other than zero cause input and/or output offsets to be added to incoming and/or outgoing data. Entry is in the form:

inp,out

where "inp" is the input offset and "out" the output offset, and should be specified directly in A/D units (e.g. -32767 to 32767 for a 16-bit A/D converter). Offset values may be positive or negative, but their magnitude should not exceed the maximum range of the A/D and D/A converters used by the hardware.

All offsets are applied to data *prior* to gain/attenuation scaling factors. This is true for both input and output; i.e. input and output scaling and offset transformations are not the inverse of each other).

In a few cases, the software may already be applying a base offset to output data *in addition* any non-zero entered output offset. This is true only for hardware that support only "offset binary" (unsigned) output. The only time this might be a problem is when the base output offset is being added by the host 80x86 processor (because the hardware doesn't have a DSP device), and thus affects the overall maximum output transfer rate. You can consult the Hardware Reference Guide to see if this is the case. If so, you can regain maximum D/A output speed by entering the negative of the default output offset and thus cancelling it to zero. But then, of course, your output data must be truly unsigned, probably requiring an intermediate Hypersignal processing step. Most hardware uses signed (2's complement) output and thus avoids the problem.

4.1.3 Sending Waveforms To DSP/Acquisition Hardware

(Real-Time Software Only)

As with acquiring data continuously in real-time from DSP/Acquisition hardware, the Analog Conversion function is used to "generate" data continuously in real-time using DSP/Acquisition hardware. The decision of which function to perform is made using the ACQUIRE/GENERATE field in the Analog Conversion function. Please consult section 4.1.2, since this field, as well as all other fields that function the same for both operations, are well-described in that section.

4.1.4 Using the ASCII Export/Import Function

The ASCII Export/Import menu selection allows waveform data files in Hypersignal binary format to be exported to and imported from files in the ASCII decimal, hexadecimal, or floating-point formats typically used with text-editors and spreadsheet programs.

This function expects all files, input and output, to be follow Hypersignal format organization and sequence (header values in a certain order, followed by data values—see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT).

NOTE: All Hypersignal waveform files have either "standard" headers or "extended" headers. In terms of integers (16-bit words), a standard header is 10 words long, and an extended header is 64 words long. However, in terms of ASCII values, which is the format of the ASCII Export/Import function, a standard header is 10 values long, and an extended header is 60 values long (because 4 values are floating-point). See chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT, for detailed information on waveform file header content and usage.

The inherent difference between the import portion of this function and the "File Acquisition" function is that the *File Acquisition function will create a Hypersignal waveform file from scratch, including the header,* by using the various data-entry parameters you give it, and also by recording facts about the data it acquires. The ASCII Export/Import function, on the other hand, can only be told to export or import, expects a header in the input file, and learns nothing about the data except its length during the process.

ASCII Export/Import Menu Fields

The following sections explain the ASCII Export/Import menu data-entry fields:

FILENAME:

Specifies the name of file to be exported or imported. In the case of export to ASCII format, the filename must have a valid Hypersignal extension as shown below. If the filename is longer than 6 characters it will be truncated to allow the data type indicator suffix to be appended. Some example export output filenames are shown below at right:

filename.TIM	Output filename will be filena_T.TXT
filename.FRQ	Output filename will be filena_F.TXT
filename.PHZ	Output filename will be filena_P.TXT
filename.PWR	Output filename will be filena_W.TXT
filename.REA	Output filename will be filena_R.TXT
filename.IMA	Output filename will be filena_I.TXT

In the case of import from ASCII format, the ASCII file must *already* be in standard Hypersignal sequence, with all header and data values in ASCII decimal, hexadecimal, or floating-point format. FILENAME must end with a valid "data type indicator", which consists of a "_X" character sequence, as shown in the example input import filenames below at left:

file_T.TXT	output binary filename will be file.TIM
file_F.TXT	output binary filename will be file.FRQ
file_P.TXT	output binary filename will be file.PHZ

file_W.TXT	output binary filename will be file.PWR
file_R.TXT	output binary filename will be file.REA
file_I.TXT	output binary filename will be file.IMA

ASCII FORMAT:

Possible entries for this field are:

'DEC'	enables ASCII decimal format conversion, (characters '0'-'9' and '-')
'HEX'	indicates ASCII hexadecimal conversion (up to 4 characters '0'-'9' and 'A'-'F')
'FP'	enables ASCII floating-point conversion (characters '0'-'9', '-','+','.', and 'E')

EXPORT/IMPORT:

Possible entries for this field are:

'E'	enables export to ASCII formats
'I'	enables import from ASCII formats

EXPORT EXPONENT EXPAND:

This field is active only for export. Possible entries for this field are:

'Y'	Expands all waveform data by multiplying by a scaling factor formed as $2^{\text{AMPLITUDE EXPONENT}}$, where AMPLITUDE EXPONENT is contained in the waveform header (see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT). If you are exporting to floating-point ASCII format, values ranging from 2.93873e-39 to 5.575e+42 can be written. For decimal or hexadecimal ASCII formats, values are rounded to the nearest integer. The exponent value written out to the exported header is then set to zero. 'Y' is the software default entry for this field.
'N'	Disables data expansion and instead exports data values directly as 16-bit mantissas, and exports the waveform header exponent value unchanged. Note that if the exponent value is zero (scaling factor equals 1), then the setting of the EXPORT EXPONENT EXPAND field will have no effect.

EXPORT HEADER:

Entering an 'N' in this field causes Hypersignal-Macro waveform header information to <u>not</u> be included in the exported ASCII file. The default entry is 'Y', which allows header information to be include. Noter: if you desire to re-import a waveform file exported without its header, <u>do not</u> use the ASCII Import function. Instead, use the File Acquisition function and set the SAMPLES/BYTES TO DISCARD field to zero.

Some Notes About Length of Export/Import Files

Export converts the entire Hypersignal binary waveform file file to ASCII regardless of whether the actual file length matches the number of samples listed in the header section of the file. All header attributes and data will be faithfully converted to ASCII. On the other hand, importing from ASCII will ask permission to continue if the number of samples listed in the header mismatches with the file length. In all cases, the number of samples attribute in the output waveform header is adjusted to exactly match the resulting file length.

The maximum number of ASCII samples that can be imported to binary file is 536,870,912. You should check beforehand, if possible, that enough disk space is available large file operations. Importing from ASCII to binary files requires disk space equal to 2 (two) bytes per sample multiplied by the total number of samples. On the other hand, calculation of the storage space needed when creating an ASCII output file may be difficult to estimate, since ASCII data is of varying length for each sample, thus making the exact file length data-dependent. A worst-case estimate would be 8 ASCII characters (or bytes) per sample (5 digits, a minus sign, a carriage-return, and a linefeed).

4.2 Displaying Time Domain Waveforms

Hypersignal contains a very general-purpose interactive display function that allows waveform display and editing, D/A output of waveform sections, printing of waveform displays, and writing sections of waveforms to new files.

Waveform display options include single, overlaid, or split-screen display, zoom, un-zoom, pan, random-access, units, amplitude scaling, axes scaling and labelling, and "active trace" selection between two displayed waveforms.

Waveform editing options include a cut-and-paste mode and a rubberband mode. In the cut-and-paste mode, sections of waveforms can be copied or deleted, and inserted or overwritten elsewhere in the waveform or into a different waveform. In the rubberband mode, individual waveform samples or sections of samples can be "shaped" by mouse movement.

D/A output can be performed to the currently selected ANALOG CONVERSION hardware in the System Configuration menu.

Waveform displays can be sent to a printer in screen-dump or publication-quality formats (the latter requires EMS memory).

Sections of waveform displays can be written to new waveform files for further processing or display.

Waveform Display/Edit Menu Fields

The following paragraphs explain how to enter the data-entry menu fields of the Waveform Display/Edit function, which, in addition to basic time domain waveform display, also serves as a general-purpose display and editor for all types of Hypersignal Waveforms.

FILENAME1:

This field specifies the name of the file containing the waveform to be displayed. Explicit entry of a waveform extension will override the default ".TIM" (time domain) filename extension—for example, a frequency domain magnitude waveform may be displayed and edited by entering its filename followed by a '.FRQ' extension.

The filename specified should be located on the current drive and subdirectory displayed in the User Setup area. The current logged drive and subdirectory can be changed at any time through use of the User Setup function under the Utilities column heading on the main menu (see section 2.5.1, Initializing the User Setup Area).

If the file does not exist on the current drive and subdirectory, or is left unspecified, an audible warning will be issued, and the cursor repositioned to the filename field. A directory of ".TIM" files (or whatever extension was entered) located on the current drive and subdirectory will appear in the System Status portion of the main menu. Paging through the directory is accomplished by using the 'PgDn' key to scroll down, and the 'PgUp' key to return to the first page of the directory. Consecutive use of the 'PgUp' key will change the search strategy for pop-up directory display—see section 3.4.2, Pop-Up Directory Display, for details.

FILENAME2:

This field specifies the name of another waveform to be dual-displayed or overlaid with FILENAME1. If this field is left blank (you can use 'Delete' or 'Backspace' keys, or press 'Enter' with the cursor at the start of the field to blank the field), only FILENAME1 will be displayed.

Normally, FILENAME2, will retain whatever entry you last put there, but is initialized to the output of a recently performed Convolution or Recursive Filtering operations to allow "comparison viewing" of filtered and unfiltered waveforms.

FRAMESIZE:

This field specifies the length of the initial display frame in samples. Initially FRAMESIZE is initialized from the FILENAME1 waveform file header (if the value contained in the header is not zero), but it can be changed at any time. Note that each time the FILENAME1 or FILENAME2 fields are entered, the framesize will be re-read from the FILENAME1 header. Minimum entry is 2; there is no maximum entry. At no time does your entry cause the FRAMESIZE parameter in waveform file headers to be modified.

LOG AMPLITUDE:

This field controls whether waveform data is displayed linearly or in log-squared format. An 'N' entry causes values to be displayed linearly. A 'Y' entry will cause waveform samples to be displayed as $10LOG(x^2)$, where x is a waveform sample, and y-axis units as "dB". You can use the AMP SCALING field below to set the maximum y-axis value for both linear and log-square display modes. Once inside waveform display, you can use the 'S' (square) key to toggle back and forth between log-squared and linear. The Hypersignal default setting for this field is 'N' (linear display mode).

Y-LABELS:

For linear display only (when LOG AMPLITUDE *field entry = 'N'),* entry in this field in the form

SCALE FACTOR,UNITS

will cause the y-axis units to be multiplied by SCALE FACTOR (a numeric value) prior to display, and UNITS (a text, or string value) to be written (labelled) vertically along the y-axis. Some entry examples:

'0.123,(Volts)'	scale factor is 0.128, units will be labelled on the y-axis as "(Volts)"
'1.789,Psi'	scale factor is 1.789, units will be labelled on y-axis as "Psi"

If you omit the UNITS portion of the entry, no label will be written. If you blank the field entirely, then the scale factor is assumed to be 1. In log-squared display mode (LOG AMPLITUDE field entry = 'Y'), the software always displays units of "dB". Regardless of the value of SCALE FACTOR, actual waveform data file is never modified because of this field.

AMP SCALING:

This field is divided into two subfields; the first specifies the maximum y-axis amplitude value for linear mode display, and the second the maximum y-axis dB value for log-squared mode display. Entries of zero cause the program to "autoscale" by using the larger of the maximum amplitude values in the FILENAME1 and FILENAME2 waveforms to be used to determine the maximum y-axis value. For example, in linear mode display, if the Y-LABELS field above had been set for Volts, then you might enter '5' in the first (linear) subfield to cause the display to show ± 5V. As another example, in log-squared mode display, you might enter '100' in the second (log) subfield to cause the display to range from 0 to 100 dB. See section 4.2.1 below, Time Domain Display Amplitude Scaling, for greater detail on time domain display amplitude scaling.

In all cases, the maximum y-axis value is multiplied by the scale factor value entered in the Y-LABELS field, above. For example, if you enter a scale factor that converts A/D counts into a physical quantity in Y-LABELS above, then you might enter a AMP SCALING value in the new units. Again, see section 4.2.1, Time Domain Amplitude Scaling, for more information.

FORMAT,PLOT:

This field specifies the format of dual-trace display and also the plot type for data sample depiction. Below are possible dual-trace entries:

'D'	dual-trace display
'S'	subtracted-trace display (assumes two-trace display)
'O'	overlaid-trace display (assumes two-trace display)

Below are possible plot-type entries:

'D'	discrete
'C'	connected
'B'	both (connected and discrete)

All combinations of the above sets of entries are valid. Of course, the FORMAT portion of this field is only active for dual-waveform display (both the FILENAME1 and FILENAME2 fields, above, have been entered).

The connected plot type utilizes linear interpolation between data samples, which might look "continuous", but is not intended to be as precise as higher interpolation orders or lowpass filter reconstruction, and is thus referred to only as "connected". The discrete plot type is the most precise, showing only actual individual sample points, but may not be as readable for large amounts of displayed data. Some examples of entry:

'D,C'	dual-trace, connected display

'O,D'	overlaid, discrete display

GRID:

This field specifies the type of grid underlay for waveform display. Possible entries for this field are:

'N'	turns off all grid display
'S'	enables solid-line grid display
'D'	enables dashed-line grid display
'I'	enables intersection grid display

Grid lines are drawn as extensions of the tic marks on the vertical axis and the major tic marks on the horizontal axis. In the case of intersection grid display, only the actual points where horizontal and vertical grid lines would intersect are displayed. Grid underlay is intended as another aide in measuring waveform amplitude levels.

4.2.1 Time Domain Display Amplitude Scaling

Default Hypersignal settings are configured to cause waveforms displayed in the Waveform Display/Edit function to have no amplitude scaling performed. Thus, waveforms are initially displayed in "A/D units" (e.g. -32768 to 32767 for a 16-bit A/D converter), and dual-trace or overlaid-trace displays perform no scaling between waveforms.

However, you have several options for amplitude scaling. First, you can change the "scaling factor" (see section 4.2, above) and the "units" label that goes with the scaling factor. For example, you might like to convert 32767 to 5 Volts, so you might enter '1.526e-4,Volts' in the Y-LABELS field. The default entry for this field is '1,Units', which means the scaling factor is initialized to 1.

Second, you might like to enter a maximum value that can be shown on the display (vertical limit). The default value for this field, the AMP SCALING field, is '0' (zero) which causes "autoscaling". Autoscaling means the maximum peak in the waveform to be used as the maximum display value. However, you might enter '20000' in this field to limit waveform display for a scaling factor set to 1. For the example scaling factor value mentioned above (1.526e-4, which converted 32767 to 5 Volts), you might set the maximum AMP SCALING value to 5.

Third, you can control scaling of dual-trace and overlaid-trace displays. The default setting is no scaling, which allows you to see relative amplitude differences between waveforms. However, if you set the DUAL AUTOSCALE field in the User Setup menu selection to 'ON', then you will always force the smaller of the two waveforms to be scaled to match the larger. This can be useful for situations where the shape is the most important consideration, and relative amplitude differences are distracting. The actual default Hypersignal setting for the DUAL AUTOSCALE field is 'FILE', which causes autoscaling for dual displays *where one or both waveforms have the "AUTOSCALE" bit in their waveform headers enabled,* and disables autoscaling in all other cases. Hypersignal does not currently set the AUTOSCALE bit for any reason, but this bit can easily be set by user-defined external programs or macro language processing. See chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT, for more information about the AUTOSCALE bit and waveform headers.

NOTE: There is currently one situation where the program automatically enables time domain display autoscaling even though the DUAL AUTOSCALE field in the User Setup menu is set to 'FILE'. This happens when one of the displayed waveforms is a quantized filter frequency response output from the Code Generate/Quantize function. See section 4.9.5.1, Simulating the Effects of Quantization, for more information. If the DUAL AUTOSCALE field is set to 'OFF' then automatic autoscaling in this situation is disabled. If the DUAL AUTOSCALE field is set to 'ON' then automatic autoscaling is always enabled.

You can achieve even more control over waveform amplitude scaling by directly changing the MAX AMPLITUDE parameter in the header of waveform files. Although not a highly recommended procedure, if you change the amplitude value in a waveform header so that it does not actively reflect the maximum absolute value that occurred in the waveform, you can "fool" Hypersignal displays into other types of scaling. For example, if the 'FILE' setting of the DUAL AUTOSCALE field is normally acceptable but you want to avoid any scaling in some cases, you could set the MAX AMPLITUDE parameter in a group of waveform headers to be identical. Then, for these waveforms only, the program will never perform any scaling of any kind since it will think they already have the same maximum absolute value.

4.3 Using Interactive Waveform Displays

The general category “interactive waveform displays” includes both waveform file displays, and analog input instrumentation displays; indeed, the line tends to blur. For example, the instrumentation displays can show a waveform file and real-time analog input at the same time. Although very different in the types of data they display, Hypersignal interactive waveform displays share much of the same user interface. This section covers the overall user interface for interactive waveform displays, and also points out cases where commands exist in one or a few types of display and not others.

Please note that there are several examples of waveform display in the Tutorial, including step-by-step menu set-ups and interactive display commands.

4.3.1 Interactive Waveform Display Commands

This section is intended as a sort of compacted “reference guide” for Hypersignal interactive waveform displays commands, including instrumentation (Digital Oscilloscope and Spectrum Analyzer) displays. Keyboard- and mouse-controlled commands that are common to all displays are covered, and some commands not found in all waveform displays are covered also.

Hypersignal waveform displays basically can be divided into 3 (three) categories:

1) analysis displays based on waveform file input (non-real-time)

2) instrumentation displays based on DSP/Acquisition hardware analog input (real-time)

3) classical and arbitrary filter graphical design (non-real-time)

All commands used in the above three types of interactive displays are listed below in alphabetical order. Although most commands are common to all three categories, a few commands may be active in only one category.

ATTACH/DETACH CURSOR: ‘Home’ key or left mouse button

In point mode only, the crosshair cursor is “attached” to the waveform; subsequent movement of the cursor “drags” waveform samples along with the cursor. If the cursor is already attached, then the cursor will be detached.

This command is active only in the Waveform Display/Edit and graphical filter editor displays. In the graphical filter editor displays, the current design method must be arbitrary to attach the cursor to the desired response.

BACKGROUND COLOR: ‘B’ key

The background color command changes the color (or intensity, blinking, or other attribute for a monochrome display) of the background used in the display. This command only affects the screen and not any images sent to a printer (the pop-up printer menu invoked by the ‘F2’ key has its own background control).

Choices can be made from a palette of 16 different colors. A background color is selected by repeatedly pressing the ‘B’ key. Each background color is displayed instantaneously. The effect of this command varies from system to system, depending on the system video display’s ability to use different colors simultaneously.

This command is not active in the Pole-Zero Plot display function, which uses the current setting of the Magnitude Display function. Also, this command is not active in the Filter Construction graphical editor displays, which use the current setting of the Waveform Display/Edit function.

COLOR:

See WAVEFORM COLOR, GRID COLOR, LABEL COLOR, BACKGROUND COLOR commands.

CONSTRUCT FILTER: ‘C’ key

This command saves the current filter design specifications, including menu parameters, and in the case of arbitrary filter designs, the desired frequency response curve, and begins the filter construction process.

This command is only active in the graphical filter editor display.

CROSSHAIR CURSOR:

See ATTACH/DETACH CURSOR, TARGET WINDOW, POINT MODE commands.

D/A OUTPUT: **'D' key**

For real-time softeware only, this command outputs the section of *the current active trace* waveform marked by the target window to the currently selected analog D/A hardware. If no target window is displayed, then the entire display frame is output to the D/A hardware. Once in progress, you can use the 'Esc' key or another 'D' key to abort the D/A output. This can be especially useful in the case of speech or music data, where features of the waveform can be isolated, and even rearranged via waveform editing (see INSERT and DELETE commands), and then listened to. The current analog D/A (signal generation) hardware is configured in the ANALOG CONVERSION field in the System Configuration menu. Output gain and base channel used are the current entries in the Analog Conversion menu.

This command is only active in the Waveform Display/Edit and 2-D Spectrograph displays.

DELETE: **'Delete' key**

In the Waveform Display/Edit function, in region mode only, this command can be used to delete or copy the *current active trace* waveform section marked by the target window to the temporary "paste" file. If the target window is not currently displayed, then the entire display frame will be operated upon. The choice between copy and delete is made by answering the prompt that appears immediately after pressing the 'Delete' key.

In graphical filter design displays, in region mode only, this command can be used to halve the density of the desired frequency response curve points in the region indicated by the target window. Decreasing density causes frequency axis points to be removed (deleted) by decimation (every other point). See sections 4.8.2.1, Using the Graphical Filter Editor, for more detail.

This command is active only in the Waveform Display/Edit and graphical filter editor displays.

ESCAPE: **'Esc' key or leftmost two mouse buttons on 3-button mouse**

Pressing the 'Esc' (escape) key at any time will return to the previous data-entry menu, and the data-entry editor will be re-activated.

DEFINE REGION:

See UNDEFINE REGION command.

FILE INITIALIZE:

See MENU INITIALIZE command.

GRID: **'G' key**

Rotates the graphical filter design grid between none, solid, intersected.

This comamnd is only active in the graphical filter editor display.

GRID COLOR: **'G' key**

The grid color command changes the color (or intensity, blinking, or other attribute for a monochrome display) of the grid and coordinate axes (if any) in the display.

Choices can be made from a palette of 16 different colors. A color is selected by repeatedly pressing the 'G' key. When first activated, a row of open squares will appear on the screen following the "G:" label at the lower left of the display. The current color selection is represented by the larger open square. For non-real-time displays, once the desired color/attribute has been selected, you must perform some command that redraws the screen (such as NEXT FRAME, BACK FRAME, REDRAW, or ZOOM) to see the color take effect. For real-time displays, the color takes effect as soon as the next frame of data acquired from the DSP/Acquisition hardware is displayed.

This command is not active in the Pole-Zero Plot display function, which uses the current setting of the Magnitude Display function. Also, this command is not active in the Filter Construction graphical editor displays, which use the current setting of the Waveform Display/Edit function.

HELP: **'?' or 'F1' keys, rightmost two mouse buttons on 3-button mouse**

Displays on-line Help pertaining to the current waveform display function. Use the 'Esc' key again to return to the current graphical display.

INSERT: **'Insert' key**

In the Waveform Display/Edit display, in region mode only, this command can be used to insert or overwrite the contents of the temporary "paste" file into the *current active trace* waveform at the point marked by the left edge of the target window. If the target window is not currently displayed then the left edge of the display is used. If the left edge of the target window is beyond the end of the waveform, then zeros are placed in the "undefined space". If no data has been previously deleted, then zeros are placed into the waveform section marked by the target window (the paste file is initialized to zeros whenever the Waveform Display/Edit function is first entered from the main menu). The choice between insert and overwrite is made by answering the prompt that appears immediately after pressing the 'Insert' key.

In graphical filter design displays, in region mode only, this command can be used to double the density of the desired frequency response curve points in the region indicated by the target window. Increasing density causes frequency axis points to be added (inserted) by linear interpolation. See sections 4.8.2.1, Using the Graphical Filter Editor, for more detail.

This command is active only in the Waveform Display/Edit and graphical filter editor displays.

LABEL COLOR: **'L' key**

The label color command changes the color (or intensity, blinking, or other attribute for a monochrome display) of all text labels in the display.

Choices can be made from a palette of 16 different colors. A color is selected by repeatedly pressing the 'L' key. When first activated, a row of solid squares will appear on the screen following the "L:" label at the lower left of the display. The current color selection is represented by the larger square. For non-real-time displays, once the desired color/attribute has been selected, you must perform some command that redraws the screen (such as NEXT FRAME, BACK FRAME, REDRAW, or ZOOM) to see the color take effect.

This command is not active in the Pole-Zero Plot display function, which uses the current setting of the Magnitude Display function. Also, this command is not active in the Filter Construction graphical editor displays, which use the current setting of the Waveform Display/Edit function.

LOG/LINEAR: **'L' key**

This command toggles both the amplitude and frequency axes between linear and log. Current axes format applies to both desired and actual (if enabled) filter responses. Four total combinations are possible. Linear amplitude (vertical axis) display may prove useful for working with frequency responses (both desired and actual) that have maximum gain greater than 1 (zero dB). This command is active only in the graphical filter editor display.

MENU or FILE INITIALIZE: **'M' or 'F' keys**

This command reinitializes the graphical filter display from the data-entry editor menu or from the specificaton file. FIR filter design files with ".FFD" extensions and IIR filter design files with ".IFD" contain both design parameters and graphical editing information. All new changes to the current edited response are lost completely when either command is executed.

This command is active only in the graphical filter editor display.

MARK SECTION: **'M' key**

Writes the current target window edges (in region mode) or crosshair cursor location (in point mode), in seconds, to a "mark" file. If in region mode and no target window is displayed, then the edges of the display are used.

Time values are always written to the mark file in pairs. If in point mode, or in region mode but the target window has no width, then the "edges" are the same, and the time values in the output pair are equal.

Mark files are stored with the currently active trace waveform filename and a ".MRK" extension, and are in ASCII format. They are intended to contain a sequence of one or more sections of interest for subsequent processing by macro source files. After being created the first time, mark files are always added to—waveform section values are not required to be in order. To re-initialize a mark file it must be manually deleted.

This command is active only in the Waveform Display/Edit display.

NEXT FRAME: **spacebar or righthand mouse button**

Pans forwards to the next display frame. For the Waveform Display/Edit function a frame refers to a number samples initially entered in the data-entry menu (the FRAMESIZE field), and altered afterwards by zoom commands. For the Magnitude and Phase Display functions, a frame refers to a number of samples initially set by the FFT length in the waveform header, and altered afterwards by zoom commands. For the 2-D Spectrograph and 3-D Spectrogram displays, which are "three-dimensional", a frame refers to an amount of time initially entered in the TIME SHOWN field on the data-entry menus for these functions, and altered afterwards by zoom commands.

A status message, "END OF WAVEFORM", will appear if you try to advance beyond the last frame of data.

This command is not active in the instrumentation, graphical filter editor, and Pole-Zero Plot displays.

OFFSET:

See VERTICAL OFFSET command.

OVERLAY: **'O' key**

This keyboard command allows the desired frequency response to be overlaid with the previous actual filter response, or to view either alone. Press the 'O' key three times to see all possibilities.

This command is active only in the graphical filter editor display.

PAN:

See NEXT FRAME, PREV FRAME commands.

POINT MODE: **'P' key**

This command puts the *current active waveform trace* into point mode. In point mode, the arrow keys or mouse motion control a "crosshair cursor" (see CROSSHAIR CURSOR, above) which can be used to provide amplitude or frequency readout. In the Waveform Display/Edit function, rubberband waveform editing can be performed in point mode. In the graphical filter editor, desired frequency response points can be rubberband-edited in point mode for *arbitrary design methods only.*

This command is active only in the Waveform Display/Edit, graphical filter editor, and 2-D Spectrograph displays.

PREV FRAME: **'Backspace' key or middle button on 3-button mouse**

Pans backwards to the previous display frame. For the Waveform Display/Edit function a frame refers to a number samples initially entered in the data-entry menu (the FRAMESIZE field), and altered afterwards by zoom commands. For the Magnitude and Phase Display functions, a frame refers to a number of samples initially set by the FFT length in the waveform header, and altered afterwards by zoom commands. For the 2-D Spectrograph and 3-D Spectrogram displays, which are "three-dimensional", a frame refers to an amount of time initially entered in the TIME SHOWN field on the data-entry menus for these functions, and altered afterwards by zoom commands.

A status message, "START OF WAVEFORM", will appear if you try to display data before the start of the waveform.

This command is not active in the instrumentation, graphical filter editor, and Pole-Zero Plot displays.

PRINT: **'F2' key**

Invokes the pop-up print menu, which allows both screen dumps and publication-quality prints of the current waveform display to nearly 200 different types of printers. The output device (LPT1, COM1, disk filename, etc.), dimensions, dots-per-inch, orientation, background, etc. can be specified in the pop-up print menu. Publication-quality prints require adequate EMS memory. Please see section 4.3.2, Printing Waveform Displays, for more detailed information.

RANDOM-ACCESS: **'F', 'T', or 'S' keys**

In the Waveform Display/Edit, Magnitude, and Phase display functions, the 'F' key will prompt you for a random-access frame value. Use the current "frame position" print-out (at upper right of waveform display) as a guide. The 'T' key will prompt you for a random-access time value. Use the current time value print-out (along x-axis for Waveform Display/Edit display, and at upper right of display, in parentheses after frame position value, for Magnitude and Phase displays). However, time entries must always be in seconds, regardless of the current display time units.

In the 2-D Spectrograph and 3-D Spectrogram displays, the 'F', 'T', or 'S' keys all will prompt you for a random-access time value. Use the current "time-spanned" print-out (at upper right of display); however, your entry must always be in seconds, regardless of the current display units.

All random-access values can be entered in integer or floating-point form (e.g. 0.34, 10.5, 415, 3.2e-3).

This comamnd is not active in the instrumentation, graphical filter editor, and Pole-Zero Plot displays.

REDRAW: **'Enter' key**

Redraws the current display using initial settings. All zoom and vertical scaling settings are reset. Display starting point is truncated to the beginning of the current waveform frame.

REGION MODE: **'R' key**

Puts the *current active waveform trace* into region mode. In region mode, the arrow keys or mouse motion controls a "target window" (see TARGET WINDOW command, below) which can be used to mark a waveform section for other commands, such as ZOOM, WRITE TO DISK, and D/A OUTPUT (real-time software only). In the Waveform Display/Edit display, cut-and-paste waveform editing can also be performed in region mode. In the graphical filter editor, desired frequency frequency points can be inserted and deleted in region mode *for arbitrary design methods only*.

This command is active only in the Waveform Display/Edit, graphical filter editor, and 2-D Spectrograph displays.

SCALING:

See VERTICAL SCALING commands.

SQUARE DATA: **'S' key**

In the Waveform Display/Edit function, this command toggles display between linear and $10LOG_{10}(x^2)$ formats.

This command is active only in the Waveform Display/Edit display.

TARGET WINDOW: **arrow keys or mouse motion**

Depending on the display and its current mode, use of the arrow keys or movement of the mouse will produce a target window or crosshair cursor.

If the display is in "region mode", then a target window will be displayed. The target window is used to "zoom" in on the waveform, and for the Waveform Display/Edit function, select sections of the waveform for cut-and-paste editing, writing sections to disk file, or sending sections to the D/A output (real-time software only). A crosshair cursor inside the target window tracks individual samples and provides an amplitude read-out at the righthand side of the display. Also, time or frequency values for the current left and right edges, width, and crosshair cursor of the target window are updated at the bottom of the display.

If the display is in "point" mode then a crosshair cursor will be displayed. Unlike the region mode crosshair cursor inside the target window, which tracks individual samples, the point mode crosshair cursor can be controlled "free-form". The point mode crosshair cursor is used to perform editing of individual waveform sample points, or produce amplitude vs. time/frequency or

frequency vs. time readout for arbitrary display locations. Also, time or frequency values for the crosshair cursor are updated at the bottom of the display.

For waveform displays that allow both region and point modes, use the 'R' key to enter region mode and the 'P' key for point mode. See also the POINT MODE and REGION MODE commands.

If the target window is displayed (region mode), then left and right arrow keys or mouse motion will translate the window left and right. Up and down arrow keys or mouse motion will make the window bigger and smaller. In all displays except 3-D Spectrogram, a crosshair cursor is also displayed inside the target window, but this is only to provide waveform amplitude tracking and readout (at right side of display) and cannot be controlled vertically (unless you change to point mode).

The target window is not active in the Pole-Zero Plot display function and the instrumentation display functions.

The crosshair cursor is not active in the Magnitude Display, Phase Display, 3-D Spectrogram, Pole-Zero Plot, or Digital Oscilloscope display functions.

TIMEBASE: **left and right arrow keys, horizontal mouse motion**

For the Digital Oscilloscope display, the left and right arrow keys (or mouse) can be used to control the current timebase setting, with the right arrow expanding the waveform (showing less time per division), and the left arrow contracting the waveform (increasing the time per division). Timebase control affects both traces in a dual-trace display; i.e. timebase control is independent of the current active trace.

This command is active only in the Digital Oscilloscope display.

TOLERANCE REGION ENABLE: **'T' key**

This command toggles graphical filter design tolerance region display on and off. Tolerance regions are always enabled upon entrance to graphical filter display for classical design methods, and disabled upon entrance to display for arbitrary design methods.

This command is active only in the graphical filter editor display.

TRACE 1 or TRACE 2: **'1' or '2' keys**

If you are in a dual-waveform display, use the '1' or '2' keys to select the "current active trace" for which other interactive commands will apply, such as crosshair cursor readout, D/A output (real-time software only), etc.

The current active trace is indicated on the display by two small rectangles at the righthand side of the display area. A filled-in (solid) rectangle indicates the current active trace. During overlaid trace display, the rectangles match their corresponding waveform colors.

This command is not active in the 3-D Spectrogram display.

TRIGGER LEVEL: **'+' and '-' keys**

Use the '+' and '-' keys to increase and decrease, respectively, the trigger level for the *current active trace.* Triggering used in the Hypersignal instrumentation displays is essentially equivalent to the "NORM" mode on a traditional analog scope, except that the last frame is frozen on the display when trigger conditions are no longer met. Note that trigger level is shown numerically on the display at the lower right as well as in the form of a "pointer" that moves up and down along the vertical axis next to the current active trace. A trigger level value of near zero effectively turns triggering off (equivalent to "FREERUN" mode on traditional analog scope).

Use positive trigger levels ('+' key) to trigger on a signal's rising edge or slope, and negative trigger levels ('-' key) to trigger on a signal's falling edge or slope. "One-shot" type display is easily achieved by pre-adjusting trigger level, then allowing input to occur. Note that on the Digital Oscilloscope data-entry menu, initial trigger level can be specified exactly in Volts.

This command is active only in the instrumentation displays.

NOTE: **Note that the current trigger trace and current active trace can be different. You might change trigger levels and wonder why they are not affecting your display—change the trigger trace to match the current active trace (see TRIGGER command, below).**

TRIGGER TRACE: **'T' key**

Use the 'T' key to toggle the current trigger trace between the displayed traces. To see—and vary (using '+' and '-' keys)—the trigger level for a trace, be sure that the active trace is the same as the desired trace. Otherwise, you can change the trigger level, and wonder why it is not affecting your display. The current trigger trace is shown at the lower right of the display.

This command is active only in the instrumentation displays.

NOTE: **The current trigger trace is completely independent of the current active trace.**

UNDEFINE or DEFINE REGION: **'U' or 'D' keys**

Undefines or defines a region currently marked by the target window (must be in region mode). Undefined regions are areas where the arbitrary design is purposefully left unspecified to the design process (to allow natural convergence in that region). The desired frequency response curve is not shown on the display in undefined regions. Defining a region simply reinstates the desired frequency response curve in that region, whether it be what the program initialized or after user-editing. Please see section 4.8.2.1, Using the Graphical Filter Editor, for more detail.

This command is active only in the graphical filter editor display.

UNITS: **'U' key**

This command rotates the current unit display (either time or frequency) up to the next standard SI value (e.g. mS to S), and then down (e.g. kHz to Hz). If unit overflow or underflow occurs, an error message is displayed and units remain unchanged. Pressing the 'U' key a third time will cause units display to be in individual time domain (sample) or frequency domain (bin) "point" units. Rotation is modulo; pressing the 'U' key four times displays all SI unit possibilities. Units will change automatically (e.g. from uS to mS) if current units are exceeded due to zoom, pan or random-access.

The UNITS command is not active in the Pole-Zero Plot or graphical filter editor displays.

VERTICAL OFFSET: **up and down arrow keys, vertical mouse motion**

Use this command to control the vertical position, or offset, of the waveform displayed in the *current active trace.*

This command is active only in the instrumentation displays. In the Spectrum Analyzer display, this command is only active when time domain data is being displayed in the current active trace.

VERTICAL SCALING: **'PgUp' and 'PgDn' keys**

The 'PgUp' and 'PgDn' keys can be used to control vertical amplitude scaling. They are active independent of other waveform display settings such as framesize, zoom, log/linear amplitude, etc. If a dual- or overlaid-trace display is active, the vertical scaling commands operate on both traces together, independent of the current active trace setting.

To scale a waveform up vertically (make it larger), use the 'PgUp' key. To scale a waveform down vertically (make it smaller), use the 'PgDn' key. Note that vertical amplitude units change along with waveform size. Also note that log mode displays (with the exception of transfer function display in the Spectrum Analyzer) "pin" the top of the display and expand/contract the data from the top down. Linear mode displays expand/contract the data about the x-axis (typically equivalent zero).

NOTE: **Waveform values exceeding the current display area vertically are not shown. They are actually considered by the program to be located accurately out of the display area, which allows that portion of lines connecting them to values within the display area to be shown with correct slope. Thus, if you see a horizontal line across the top of the display area connecting two or more waveform samples, then these samples are actually of that value. In this way, "clipped" input, or other "saturated" input can be easily discerned from input that is actually beyond the current screen limits.**

WAVEFORM COLOR: **'C' key**

The waveform color command changes the color (or intensity, blinking, or other attribute for a monochrome display) of the waveform in the display.

Choices can be made from a palette of 16 different colors. A color is selected by repeatedly pressing the 'C' key. When first activated, a row of solid squares will appear on the screen following the "C:" label at the lower left of the display. The current color selection is represented by the larger square. For non-real-time displays, once the desired color/attribute has been selected, you must perform some command that redraws the screen (such as NEXT FRAME, BACK FRAME, REDRAW, or ZOOM) to see the color take effect. For real-time displays, the color takes effect as soon as the next frame of data acquired from the DSP/Acquisition hardware is displayed.

This command is not active in the Pole-Zero Plot display function, which uses the current setting of the Magnitude Display function. Also, this command is not active in the Filter Construction graphical editor displays, which use the current setting of the Waveform Display/Edit function.

WINDOW:

See TARGET WINDOW, ZOOM commands.

WRITE TO DISK: **'W' key**

In the Waveform Display/Edit display, this command writes out to a disk waveform file the section of the *current active trace* waveform marked by the target window. If no target window is displayed, then the entire display frame is written to the output file.

After initially striking the 'W' key, you will be prompted for an output filename. If the output file already exists, you will be prompted whether to overwrite the existing file.

Output waveform file header attributes are copied from the display file, except for number of samples and max amplitude, which are updated as necessary.

NOTE: **In the instrumentation displays, this command writes out to a disk waveform file the current active trace waveform, whether it be analog input or waveform file data. The Digital Oscilloscope function always writes out time data (".TIM" filename extension). Possible Spectrum Analyzer output formats include time data (".TIM" filename extension), magnitude data (".FRQ"), phase data (".PHZ"), and power spectra data (".PWR"). For the instrumentation displays, there are many exceptions and additions to the above description; please see section 4.7.2.11, Summary of Interactive Instrumentation Display Commands, for fully detailed information.**

This command is active only in the Waveform Display/Edit and instrumentation displays.

ZOOM: **'Home' key or left mouse button**

In region mode only, the section of the *current active trace* waveform inside the target window is expanded to become the full waveform display.

This function is not active in the graphical filter editor, Pole-Zero Plot, and instrumentation displays.

ZOOM-UNDO: **outer two mouse buttons**

In region mode only, the display returns to the display previous setting before the last ZOOM command. This command is a sort of "zoom-out", allowing convenient temporary zoom without having to return to the data-entry menu or REDRAW the entire waveform display. Sixteen (16) levels of "zoom history" are kept.

This function is not active in the graphical filter editor, Pole-Zero Plot, and instrumentation displays.

4.3.2 Printing Waveform Displays

In order to send waveform displays to various printers and disk files, Hypersignal uses a built-in print function. This function can perform both "screen dumps" and "publication-quality" prints. A screen dump is defined as simply transferring the current contents of video RAM directly to the printer, without any increase in resolution. Of course, for printers with resolution greater than about 90 dots per inch, this will not fully utilize the capability of the printer. A publication-quality print is defined as regenerating the current screen display in EMS memory, at much higher resolution, and then transferring this image to the printer.

The Hypersignal print function can be invoked form all waveform display functions, including graphical filter design and instrumentation displays (Spectrum Analyzer and Digital Oscilloscope).

After being invoked via the 'F2' key from inside any Hypersignal waveform display, the print function will display a pop-up menu asking you for the type of printer (nearly 200 are supported, including LaserJet and PaintJet), the output device (LPT1, COM1, disk filename, etc.), the dots-per-inch (dpi) value for the printer (the "xres" and "yres" fields), the desired dimensions of the print (the "xlen" and "ylen" fields, entered in inches), image scale factors ("xscale" and "yscale"), the page orientation ("portrait" or "landscape"), the type of background ("white" or "black"), the number of copies, and whether the operation is to be a screen dump or publication-quality print.

If you are performing a publication-quality print, the total amount of EMS memory needed can be approximately calculated as follows (in kBytes):

$$\frac{(\text{xlen inches} * \text{xres dpi}) * (\text{ylen inches} * \text{yres dpi}) * (\text{bits-per-pixel})}{8}$$

"Bits-per-pixel" is one (1) for most single-color ("black and white") printers, and is LOG_2(number of colors) for printers capable of printing in color. For example, the Hewlett-Packard PaintJet can print 16 colors, so its bits-per-pixel value is four (4).

NOTE: **One exception occurs for black-white printers when printing from the 2-D Spectrograph function: "simulated" gray-scale operation causes memory requirements to be the same as if for a 16-color printer.**

If you receive an error message of the form "...not enough EMS memory..." when attempting a publication-quality print, try reducing the dimensions (inches) specified in the pop-up menu. Also make sure you have the scale factors set to 1 (one); they are really only intended for screen dump operation anyway. Finally, make sure that some other memory-resident program is not using EMS memory (also make sure the Hypersignal overlay file has not been loaded into EMS memory via the '/E' command-line switch).

If you don't have any EMS memory (in which case you should receive an error message of the form "...EMS memory not found or not installed..."), then you must use the screen dump option. In this case you must control the dimensions of the output image by use of the "scale factors". Since the number of total pixels is fixed by the current video hardware, the dimension fields ("xlen" and "ylen") become useless. For example, if you enter a printer dpi ("xres" and "yres") of 300, and your video resolution is 640x480, then your resulting image will only be about 2 x 1.5 inches unless you "scale it up" with the x and y scale factors. However, when you apply scale factors greater than one, remember that image resolution doesn't truly change; only image *size* changes (the image gets bigger but it also gets "blockier").

Of course, if you desire enough resolution to put the Hypersignal waveform display in a paper or report, then EMS memory becomes a must. The improvement between screen display and actual print-out at very high resolutions (2000x2000 pixels or greater) is quite remarkable, being achieved not by "smoothing" or "convolution" techniques performed on existing video RAM memory, but by actually re-generating the entire image *from scratch* in the temporary EMS memory area, and then sending this huge image to the printer. EMS memory is available in many forms, ranging from an add-in board or clone to fast, extra motherboard memory provided by the PC manufacturer. Note that whatever type is used, *you must have an EMS driver installed*

before running Hypersignal (should be listed in the CONFIG.SYS file on the root directory of your system) that adheres to the LIM (Lotus-Intel-Microsoft) standard. The driver should be provided by the machine or motherboard manufacturer.

NOTE: EMS memory re-draws and subsequent prints *will* require a long time. If your image is big enough, and/or in gray-scale or color, you may want to go to lunch or something. Even screen dumps with scaling factors greater than one or the smoothing filter enabled can take quite some time.

You must have access to the file "BDRV.PDT" (printer-driver definition table), located either on the path containing the "HSxxxxx.EXE" file, on the current DOS path, on the current Hypersignal User Setup path, or in the DOS path list.

If you have a printer not currently listed in the printer-driver definition table, or wish to change the default conditions of an existing printer, you can run the utility "BDE.EXE" (which requires access to the "BDRV.PDT" file on the same subdirectory). This utility is the printer-driver definition table editor, which allows new printer-driver definitions to be added, and existing ones to be modified.

NOTE: When you perform a print to a black and white printer from a color waveform display, the program will automatically translate all colors to contrast against the specified background. In the case of a screen dump, this causes the screen display to temporarily show all colors as white on black, then return to its initial colors. However, in one case, the special 3-D map mode of the Spectrum Analyzer function, the program is unable to correct for colors that will not print on black and white printers, as it is able to do in all other cases. This is due to the fact that there is no way to store the huge amounts of waveform data that 3-D map mode may display on one screen for purposes of later recall and re-draw in video memory using "printable" colors. Thus, you must manually set white waveform, grid, and text color selections for expected 3-D map mode print-outs.

4.3.2.1 Using the Pop-Up Print Menu

After invoking the print function inside a Hypersignal display via the 'F2' key, a pop-up menu containing image and printer parameters will appear.

Use the left and right arrow and carriage return ('Enter') keys to move from field to field. The up and down arrows *will not move from field to field*, but will instead change the value of the field in prescribed (suggested) increments. For example, you may use the up and down arrow keys in the "printer type" field to search through the entire printer-driver definition table, or in the "xlen" and "ylen" fields to set the horizontal and vertical dots-per-inch to acceptable values. Of course, you can always enter any field as you wish, but some entries have only limitied possibilities and will take on their nearest acceptable value at print-time. As in all other Hypersignal data-entry menus, pressing <ctrl-Enter> or entering the last field will start the print function. Prior to this, the 'Esc' key will return to the current display function.

Below is an explanation of each field in the Hypersignal waveform display pop-up printer menu:

printer type:

Number and name of output printer. Even when printing to a file you should still select the correct eventual printer when generating the file. Use the up and down arrow keys to browse through the supported printers.

output:

Either a device or file. Filenames are pre-set to the name of the primary waveform filename, plus a ".TI0", ".FR0", ".PH0", etc. extension, where the zero has been added to distinguish print files from actual waveform files. The "TI", "FR", "PH", etc. come from, of course, the "TIM", "FRQ", "PHZ", etc. extensions used elsewhere in the program. Use the up or down arrow key to review all possible output options.

xres:

Horizontal dots-per-inch for the printer. *Try and use only those values allowed by your printer (consult your printer documentation).* Initial values shown when you first enter a printer type (see above) should be legitimate; increasing/decreasing "xres"

with the up and down arrow keys doubles/halves “xres”, which is also usually legitimate. However, printers are strange beasts, and it is highly recommended that you check your printer manual before changing the default entry.

yres:

Same as “xres” above, but for vertical direction. Note that frequently printer “xres” and “yres” values are the same—but not always. Again, please check your printer manual.

xlen:

Width in inches of printed image when performing an EMS memory regeneration. This field is basically useless when doing a “screen dump” instead of an EMS memory regeneration (see discussion above).

ylen:

Height in inches of printed image; see “xlen” above.

xscale:

Horizontal scale factor when converting the RAM image (either EMS memory or video memory) to a printed image. This field is very useful when performing screen dumps, providing more or less your only control over printed image dimensions, since the number of pixels is fixed by the current video hardware. See the discussion in the previous section and section 3.5 about scale factors.

For EMS memory regenerations, the scale factors are generally not used; they should be set to 1 (one) in order to fully control the final dimensions of the printed image using the “xlen” and “ylen” fields (which are specified in inches).

yscale:

Same as “xscale” above, but for vertical direction.

Portrait/Landscape:

Use ‘P’ to send images to the printer oriented as on the screen. Use ‘L’ to rotate images 90 degrees clockwise before printing. Landscape may be useful when trying to maintain the same dimensions and aspect ratio of the screen, but the screen width is larger than the paper width.

background:

Use ‘W’ to print images and leave “white” paper showing as the background. Use ‘B’ to print images that are faithful to the screen, in the sense that the screen is normally “black”. ‘B’ is most often used when using a color printer, where colors such as yellow and white either wash out or don’t show at all unless a black background is used.

When a ‘W’ is selected in this field a true white color on the screen prints as black on the printer. When a ‘B’ is entered, white on the screen is left untouched (white) by the printer.

copies:

Number of copies desired for current print.

EMS Memory Regenerate/Screen Dump:

‘E’ specifies a complete image regeneration in EMS memory at resolutions and dimensions controlled by the “res” and “len” fields, respectively. If not enough EMS memory is available to perform the specified print, then an error message will be displayed. ‘S’ specifies a screen dump from video memory at resolutions/dimensions controlled by the “res” and “scale” fields. Entering ‘SS’ will enable a smoothing filter that may be useful for “scale” values greater than one (although *greatly* increasing image processing/printing time). See the initial discussion in section 3.5, above, concerning EMS memory regeneration vs. screen dumps.

4.3.3 D/A Playback From Waveform Displays

(Real-Time Software Only)

Inside the Waveform Display/Edit and 2-D Spectrograph display functions, the current section of the waveform indicated by the target window (see ZOOM command in section 4.3.1, Interactive Waveform Display Commands) can be sent to the D/A output of the currently configured DSP/Acquisition hardware. The currently configured DSP/Acquisition hardware for this function is set in the ANALOG CONVERSION field of the System Configuration menu.

If a target window is not displayed, then the entire display frame or segment is sent to the D/A output. If a dual-trace or overlaid-trace display is active, then waveform data is output from current active trace (and its corresponding sampling frequency) is used. Use the '1' and '2' keys to select the current active trace (note the trace-indicator rectangles at the righthand side of the display).

D/A output can be invoked at any time during waveform display. For example, in the Waveform Display/Edit function you could use the D/A output function to listen to sections of a speech waveform at any time during a waveform editing section.

The output gain and analog output channel used are the current entries in the Analog Conversion menu.

4.3.4 Editing Waveforms

Waveform editing inside Hypersignal can be performed in the Waveform Display/Edit function, in two different modes: cut-and-paste, and rubberband. Note that the Waveform Display/Edit function can be used to display and edit any waveform, not just time domain waveforms. To work with other types of waveforms, enter their filename extension on the Waveform Display/Edit data-entry menu in the FILENAME1 or FILENAME2 fields.

Cut-and-paste editing usually relies on the target window (see ZOOM command in section 4.3.1, Interactive Waveform Display Commands) to indicate waveform sections that should be "cut" (deleted or copied) and then later "pasted" (inserted or overwritten). However, if no target window is displayed, then the section size is assumed to be the entire display frame. If dual-trace or overlaid-trace display is active, then the '1' and '2' keys can be used to select which trace is active at any point (allowing transfer of data between waveforms of between channels of a single waveform). Cut-and-paste editing is performed in "region mode" (default mode). The 'R' key puts the Waveform Display/Edit display into region mode.

Rubberband editing editing is performed in "point mode" ('P' key). When in point mode, a crosshair cursor is displayed that can be moved about freely on the display (as opposed to the cursor in region mode, which "tracks" individual waveform values). By pressing the 'Home" key to "attach" and "detach" the crosshair cursor from the waveform, you can use the mouse or arrow keys to drag waveform points around. This is sometimes referred to as "rubberbanding", since waveform points on either side of the cursor stay fixed, giving the impression of stretching the waveform. By then moving the cursor from side-to-side, and depending on your level of zoom into the waveform, you can easily change small or large amounts of the waveform into a desired shape.

Cut-and-paste and rubberband editing of waveforms may be freely intermixed in the same editing session. Use the 'R' key (region mode) and the 'P' key (point mode) to switch from one mode of editing to the other.

NOTE: **Any time a waveform has been edited in the Waveform Display/Edit function, then upon exiting the Waveform Display/Edit data-entry menu or attempting to display a different waveform, you will be prompted whether or not to save the modified waveform. If you answer 'Y', the modified waveform will be saved and the previous, unchanged waveform saved with a ".BAK" filename extension. If you answer 'N', all editing changes will be lost.**

4.3.4.1 Cut-and-Paste Waveform Editing

While in region mode (use 'R' key), and after positioning the target window using the arrow keys or mouse, you can delete or copy sections of the current active trace waveform into the "paste" file. If the target window is not displayed, the software will perform the delete or copy operation on the entire display frame. The paste file is kept until you return to the main menu. This means that even if you "escape" to the Waveform Display/Edit data-entry menu, and then re-enter the waveform display, the paste file retains its previous contents. To delete or copy a waveform section, first press the 'Delete' key. You will then be prompted to press the 'D' or 'C' keys, after which both the paste file and waveform display will be updated.

Conversely, you can insert or overwrite the current paste file contents into the current active trace waveform. First, position the *left edge* of the target window at the point where you would like to insert/overwrite the contents of the paste file. If the target window is not displayed, the software will perform the insert or write operation assuming the left edge of the current display. Then, press the 'Insert' key. You will then be prompted to press the 'I' or 'W' keys, after which the waveform display will be updated. After an insert or overwrite operation, the paste file contents remain unchanged. Upon first entering the Waveform Display/Edit function, the paste file is initalized to zeros. If you insert or overwrite at this point, without first deleting or copying a waveform section, zeros will be inserted into the waveform. This is the only case where the size of the target window is used to determine the amount of data inserted or overwritten; in all other cases, the entire paste file contents determine the size of the insert or write.

Note that cut-and-paste operations performed on stereo or complex waveforms will affect both traces, regardless of which trace is currently active, since these waveforms are stored in an interleaved manner.

If you perform an insert or overwrite operation beyond the end of a waveform, zeros will be inserted between the end of the waveform and the start of the new data.

Some suggestions for exact cut-and-paste editing:

1) Consider enabling "discrete" or "both" (connected and discrete) display before editing—it will be easier to see individual samples. This is important, because the target window edges are displayed *exactly on top of samples*; thus, all deletes and copies include the target window edges. To delete a single sample, you must reduce the target window to "zero width", or a single vertical line, placed directly on top of the desired sample to delete.

2) Zoom to an appropriate level before editing, again so you can see individual samples.

3) Always use the "L:" and "R:" time value displays at the bottom of the display when needing to position the target window exactly.

4) Don't use the target window at all; instead rely on FRAMESIZE entry in the data-entry menu and the random-access time prompt ('T' key) to exactly position sections of data to occupy the entire display frame prior to cut-and-paste operations.

Below is a summary of the four possible editing commands when using cut-and-paste waveform editing:

DELETE: **'Delete' key, followed by 'D' key**

Waveform data inside target window read into "paste" file, then deleted from waveform, waveform is re-displayed. If the target window is not displayed, then the entire display frame is deleted.

COPY: **'Delete' key, followed by 'C' key**

Waveform data inside target window read into "paste" file, waveform remains unchanged. If the target window is not displayed, then the entire display frame is copied.

INSERT: **'Insert' key, followed by 'I' key**

Waveform data currently in "paste" file inserted into waveform starting at left edge of target window (i.e. current window width irrelevant), waveform re-displayed. If the target window is not displayed, then the current "paste" file data is inserted at the left edge of the display.

OVERWRITE: **'Insert' key, followed by 'W' key**

Waveform data currently in "paste" file overwritten onto waveform starting at left edge of target window (i.e. current window width irrelevant), waveform is re-displayed. If the target window is not displayed, then the current "paste" file data is overwritten starting at the left edge of the display.

4.3.4.2 Rubberband Waveform Editing

While in point mode (use 'P' key), you can freely position the crosshair cursor anywhere in the current active trace of the waveform display, and use the 'Home' key (or left button on mouse) to attach and detach the cursor from individual waveform samples.

Once attached to the waveform, continued movement of the crosshair cursor will “drag”, or “rubberband” the waveform values. The new waveform shape is drawn in a different color. Note that, depending on your current level of zoom, you can change just one sample or thousands of samples with the same mouse movement.

Any command that causes the display to be re-drawn (zoom, pan, etc.) will show your recent point mode editing changes as part of the current waveform, using the same color. Remember that you must first switch to region mode (use the ‘R’ key) in order to perform a zoom, which requires the target window instead of the crosshair cursor.

Rubberband editing can be performed beyond the end of the currently displayed waveform, in which case the software will insert zeros in unedited areas after the end of the existing waveform data.

Note that rubberband editing can be performed on a single trace of stereo or complex waveforms, if desired. To edit both traces identically, you must repeat each editing sequence.

One suggestion for exact point mode editing:

First position the crosshair cursor *exactly* where you want it, then press the ‘Home’ key to move the waveform, then immediately press the ‘Home’ key again to avoid unwanted waveform movement. Repeat the process for exact positioning of a sequence of waveform samples.

4.3.5 Writing Sections of Waveforms To a New File

Sections of an existing waveform can be written to a new waveform file from inside the Waveform Display/Edit display function. Note that the Waveform Display/Edit function can be used to display any waveform, not just time domain waveforms. To work with other types of waveforms, enter their filename extension on the Waveform Display/Edit data-entry menu in the FILENAME1 or FILENAME2 fields.

After positioning the target window (use arrow keys or mouse), you can press the ‘W’ key to cause the waveform section for the current active trace to be written to a file. You will first be prompted for the name of the new output waveform—do not enter an extension, as the new waveform will inherit the extension of the current displayed waveform. If the target window is not active (not showing), then the entire display frame will be written to the new file.

4.4 Frequency Transformation (FFT Generation)

The FFT Generation menu selection allows a complex Fast Fourier Transform (FFT) to be performed on a time domain input waveform.

The FFT Generation function has many options which may be user-specified, including framesize, FFT length, and overlap, which allow for combinations of “zero-filling” and overlap. Popular time domain windows may also be applied prior to FFT processing. Also, if a supported DSP board has been selected in the DSP ACCELERATOR field of the System Configuration menu, then the each frame of time domain data will be downloaded to the DSP board, where the actual processing will take place, resulting in an increase in processing speed.

If the input time domain waveform is non-complex, then the imaginary input data to the FFT function will be zero-filled. Input waveforms can be made complex by storing data in the correct format and setting the “complex” attribute in the waveform header. This is possible with the File Acquisition and Difference Equations functions, or by creating the waveform external to Hypersignal. See chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT, for more information about complex waveforms.

FFT Generation Menu Fields

The following sections explain the FFT Generation menu data-entry fields:

FILENAME:

This field specifies the name of the file containing the input time domain waveform. If not found, a directory of “.TIM” files will appear in the System Status area; use the ‘PgUp’ and ‘PgDn’ keys to search for a waveform (see section 3.4.2, Pop-Up Directory Display).

FFT ORDER:

This field specifies the order of the complex FFT. The length of the FFT = $2^{\text{FFT ORDER}}$. The maximum FFT order is 12; the minimum is 2. For a discussion on zero-filling and truncation, see section 4.4.2 below.

This field is normally initialized from the FFT order value inside the input waveform header, unless this value is zero.

In some cases where a DSP ACCELERATOR has been configured (System Configuration menu), the current DSP board may not have enough memory to support the requested FFT length. In such cases, Hypersignal will query you as to whether to continue using the 80x86 and/or 80x87 coprocessor (which is equivalent to temporarily setting the DSP ACCELERATOR field to 'none').

FRAMESIZE:

This field specifies the length of each input time waveform frame in samples. If the framesize is less than the FFT length (see above), then the remainder of the frame will be zero-filled. If the framesize is greater than the FFT length, then a prompt will appear whether to truncate FRAMESIZE to the FFT length or increase the FFT length to encompass the framesize. For a discussion on zero-filling and truncation, see section 4.4.2 below. For some suggestions on choosing an appropriate value of framesize for different waveforms and types of data, see section 4.4.1 below.

This field is normally initialized from the framesize value inside the input waveform header, unless this value is zero.

% FRAMESIZE OVERLAP:

Amount of overlap between successive frames of the input time waveform. Overlap can be pictured as samples at the end of one time frame and the beginning of the next that are in common, or used more than once, when the frames are input to the FFT function. Minimum overlap is zero, in which case no samples are in common between successive time frames; maximum overlap is one less than FRAMESIZE, in which case all samples but one are used more than once.

Essentially, use of overlap increases the number of FFT frames output for a given amount of time, producing increased "time resolution." This can be useful when displaying the results using the 3-D Spectrogram or 2-D Spectrograph functions, which depict frequency and amplitude versus time.

See section 4.4.1 below, Framesize and Overlap Discussion, for greater detail on choosing appropriate FRAMESIZE and OVERLAP values.

WINDOW:

This field specifies the type of window to be applied to each input frame of the time waveform prior to being FFT-processed. Possible entries are:

'R'	rectangular (i.e. none)
'Hm'	Hamming
'Hn'	Hanning
'Bl'	Blackman
'Bt'	Bartlett

Windows are always calculated as being FRAMESIZE (above) samples long and applied to each frame of the input data symmetrically. Thus, if FRAMESIZE is less than the FFT length, the zero samples beyond FRAMESIZE are not included in the windowing process. Also note that when FRAMESIZE is even, the actual "center point" of the window lies on a "half-sample", causing the points FRAMESIZE/2 and FRAMESIZE/2+1 of the window to be equal.

REAL/IMAG OUTPUT:

A 'Y' entry directs intermediate complex results to be stored in output files, at the cost of increased FFT processing time disk storage space. Real and imaginary output is *necessary for subsequent inverse FFT operations.*

Note that real and imaginary files may be manipulated inside the "Difference Equations" function by explicitly specifying ".REA" and ".IMA" extensions after the filename in the x1 and x2 external waveform fields.

PRECISION:

> **NOTE:** **This field is not active (does not appear on the menu) if a DSP-based hardware is currently selected in the DSP ACCELERATOR field of the System Configuration menu.**

When DSP hardware is used to perform the FFT processing, the precision used is the natural word-length of the DSP chip. For example, fixed-point DSP hardware might use 16-bit or 24-bit precision. Floating-point DSPs normally perform all FFT processing using 32-bit floating-point calculations. However, if the DSP ACCELERATOR field is set to 'none', then the PRECISION field is active and the following entries are possible:

'R' Specifies 8-byte floating-point precision (unless FFT size is 4096, in which case 4-byte floating-point precision is used).

'I' Specifies integer precision (16x16 multiply, 32-bit add/subtract, and 32x16 divide/shift). Integer precision invokes 80x86 assembly language routines, and provides greatest speed, but with greater chance of numerical overflow.

If overflow is detected, the program will automatically rescale the input data by 0.5 and re-start, repeating this process as necessary to achieve no overflow.

The actual input waveform file is never modified.

Overflow using real precision can occur only if display results cannot be stored with integer precision, in which case scaling will again be automatic, repeating as necessary.

Should any automatic scaling of input data occur during FFT processing operations, the resulting scale factor is stored inside the output frequency domain file waveform headers, where it is then applied during frequency domain display and processing operations.

It is strongly suggested that precision not be mixed between forward and subsequent inverse FFT operations—especially integer forward FFT and real inverse FFT, which may produce a substantial loss of accuracy in reproducing the original time domain waveform. Thus, it is suggested that when future inverse FFT operations are anticipated, real precision be used if possible. For most operations, use of a floating-point DSP accelerator qualifies as "real precision".

MAGNITUDE/PHASE APPROXIMATION:

> **NOTE:** **This field is not active (does not appear on the menu) if a DSP-based hardware is currently selected in the DSP ACCELERATOR field of the System Configuration menu.**

However, if the DSP ACCELERATOR field is set to 'none', then this field is active and a 'Y' entry in this field invokes magnitude and phase approximations for frequency domain output. These approximations avoid squareroot and ARCTAN functions, thus providing greater speed in FFT processing time, but at the cost of significantly less accuracy in subsequent magnitude (especially in high-amplitude areas) and phase display. Approximations used are:

$$\text{mag}[x, yj] = 0.3364 * \text{MIN}(|x|,|y|)+\text{MAX}(|x|,|y|)$$

$$\text{phase}[x, yj] = A - z*(0.971899 - 0.191848*z*z)$$

where for $|y/x|<1$, $z=y/x$, $A=0$;
and for $|y/x|\geq 1$, $z=x/y$, $A=\pi/2$ if $x/y\geq 0$, $A=-\pi/2$ if $x/y<0$

It is strongly suggested that when real precision is selected (PRECISION field, above), the MAG/PHASE APPROXIMATION field be set to 'N', in order to take full advantage of the extra precision. The speed penalty incurred by not using magnitude and phase approximation is negligible once you have decided to accept the lower processing speeds of real precision.

4.4.1 Framesize and Overlap Discussion

Below are brief suggestions for selecting the FRAMESIZE and OVERLAP fields used in the FFT Generation and LPC Autocorrelation functions, as they sometimes tend to cause confusion.

Generally, FRAMESIZE should be chosen such that you feel your data remained "quasi-stationary" for this amount of samples (a duration of time equal to FRAMESIZE divided by the sampling frequency of the waveform). By quasi-stationary, it is meant that the major frequency components in the data didn't change much during that period of time; thus, by segmenting the input waveform into successive sections of FRAMESIZE length, changing frequency characteristics can be viewed versus time in the 3-D and 2-D frequency displays.

Once a FRAMESIZE value has been chosen, it is usually a good idea to choose a WINDOW type to reduce or eliminate any "edge" noise due to the segmenting of the input waveform data into arbitrary frames (the Blackman window is one that tapers smoothly to zero at both ends of the FRAMESIZE length).

If a non-rectangular window is to be used, then it is also usually a good idea to use at least 50% OVERLAP. This will ensure that noise spikes, "glitches", or other short (down to single-sample duration) data will show up clearly in the 3-D and 2-D frequency domain displays (even if they occur near the frame edges, where the window values are small).

Note that if you are choosing FRAMESIZE and OVERLAP values for the LPC Autocorrelation function, then subsequent FFT analysis of the LPC Autocorrelation output *should use the* LPC ORDER+1 *as a FRAMESIZE value, rectangular window, and no overlap;* LPC waveforms are initialized this way by the FFT Generation function automatically. If you also process *the unautocorrelated waveform* through the FFT Generation function, you should use the same values of FRAMESIZE, OVERLAP, and WINDOW that were used in the LPC Autocorrelation function. This will allow you to make exact comparisons in the frequency domain between the autocorrelated and unautocorrelated waveforms.

4.4.2 Zero-Filling and Truncation Discussion

In the FFT Generation function, when you specify a framesize less than the FFT length (FFT length = $2^{\text{FFT ORDER}}$), you will have zero-filling. That is, the remainder of the frame, until the full FFT length, will be zero-filled. Zero-filling does not change the resulting magnitude of the output frequency domain waveform, and can be useful for increasing the "frequency resolution" for the given amount of time represented by the framesize. Increased frequency resolution can allow you to more accurately locate energy in the frequency domain, since there is fewer Hertz between each frequency point. Also, increased frequency resolution can help with phase-unwrapping, in which case more frequency points can mean less ambiguity in areas that have noisy phase.

When you specify a framesize greater than the FFT length, you will have truncation. Truncation simply means that the effective framesize used for the FFT Generation function will be less than what you first specified. Normally this is only a problem if the input time domain waveform was of a type where a certain amount of data was *required* to be in one frame. This is the case with filter impulse responses—splitting these waveforms into two or more frames will produce frequency domain results that inaccurately depict the filter's response across the entire frequency range (from dc to half the sampling rate). Another case would be the input and output waveforms used for transfer function analysis—these also have to be contained in a single FFT frame.

4.4.3 Estimating Power Spectra

Hypersignal contains a frequency domain function, Power Spectra Generation, that operates on frequency domain magnitude waveforms. This function allows three different types of estimation to be performed, creating a Hypersignal "power spectra" waveform file (".PWR" filename extension) that can be displayed and manipulated in all frequency domain displays in the same manner as magnitude waveform files (".FRQ" filename extension).

Power spectra waveform files are calculated essentially from one or more input magnitude frames averaged together or filtered producing one or more output frames (sometimes called periodograms). The number of frames to average or filter is specifiable in the data-entry menu. Output is always "root-mean-square"; in other words, output is calculated as the square of the specified number of magnitude frames summed together (or filtered, depending on method of averaging), divided by the number of frames, and then raised to the 1/2 power.

To see power spectra generation results, use any of the Magnitude Display, 3-D Spectrogram, 2-D Spectrograph, or Spectrum Analyzer functions and enable the data-entry menu POWER SPEC field, where a 'P' entry enables power spectra generation output display ('M', the default, enables standard magnitude display). All options and interactive commands for these displays, including dual- or overlaid-trace display, function with power spectra input exactly as with magnitude input.

Power Spectra Generation Menu Fields

The paragraphs below explain the data-entry fields in the Power Spectra Generation menu selection:

FILENAME:

This field specifies the filename of an input frequency domain magnitude waveform (".FRQ" filename extension) from which power spectra will be calculated, and the filename of the output frequency domain power spectra waveform (".PWR" filename extension) that will be created. The filename specified should be located on the current drive and subdirectory displayed in the User Setup area. The current drive and subdirectory can be changed at any time through use of the User Setup function under the Utilities column heading on the main menu (see section 2.5.1, Initializing the User Setup Area).

If the waveform file does not exist on the current drive and subdirectory, an audible warning will be issued, the cursor repositioned to the filename field, and a directory of ".FRQ" files will be displayed. Paging through the directory is accomplished by using the 'PgDn' key to scroll down, and the 'PgUp' key to return to the first page of the directory. Consecutive use of the 'PgUp' key will change the search strategy for pop-up directory display—see section 3.4.2, Pop-Up Directory Display, for details.

METHOD:

This field specifies the practical method to use when estimating power spectra. Possible entries are as follows:

'A' — Causes power spectra to be estimated by averaging successive groups of FFT data frames together; the number of FFT data frames in the groups set by FRAMES TO AVERAGE, below, which is interpreted as a percentage of the total number of input waveform frames. Thus, an entry in the FRAMES TO AVERAGE field of '100' would specify all frames to be averaged together, resulting in one output frame. This method is equivalent to the Bartlett power spectrum estimate if no time domain window was specified in the original FFT function, and to a Welch estimate if a non-rectangular time domain window was specified.

'RA' — Enables estimation by a running average, which, like straight averaging above, uses FRAMES TO AVERAGE as the number of frames to average together at any one time, but, unlike straight averaging, advances only one FFT data frame at a time, instead of averaging one group and then advancing to the next. For example, for a frequency domain file containing 32 FFT data frames, setting FRAMES TO AVERAGE equal to 8 would result in 4 power spectra frames for straight averaging, but would result in 40 power spectra frames using a running average (the extra 8 frames are added to the end of a running average to allow it to "die out").

'E' — Enables an exponential filter technique, where a one-pole lowpass filter is applied using FFT data frames as input, and power spectra output in the form:

$$y[n,k] = \sqrt{(A*x[n,k]^2 + B*y[n-1,k]^2)}$$

for k=0,1..N-1, where N is FFT length, y[n,k] is power spectra output at time n and frequency index k, and x[n,k] is FFT data frame input. A and B are the one-pole filter coefficients, where A is calculated as 1/(FRAMES TO AVERAGE) and B as 1-A, so as to maintain unity gain.

FRAMES TO AVERAGE:

This field specifies the value used as the number of frames to average; its use is dependent on the METHOD selected (see above). Note that this field is interpreted as a percentage when straight averaging is used ('A' entry in the METHOD field).

4.4.4 LPC Autocorrelation

Although performed in the time domain, linear predictive coding (LPC) is often used to provide a simplified picture in the frequency domain. Basically, when given an input waveform, linear predictive coding attempts to derive a set of filter coefficients, that when applied to an impulse, or impulse train, will create a "synthesis filter" response that approximates the original waveform. Characteristics of this type of analysis include:

1) Compression. Storage or transmission of the filter coefficients might achieve a tremendous savings compared to using the total information in the original waveform.

2) “Filtered” view in the frequency domain. Only the frequency areas that contain significant energy will be present in an FFT analysis of the filter coefficients.

3) The technique works only with periodic or semi-periodic data, since the filter response must basically oscillate or “ring” in some fashion to create a waveform in this manner. In fact, the technique is sometimes used to indicate the “randomness” of input data: autocorrelation of truly random data will produce virtually a zero result, since sequences of a random waveform cannot be expected to correlate with one another.

4) Subsequent frequency domain display, after FFT Generation of the output filter coefficients, must be performed using the “negative log” format. This is the default initialization used by all Hypersignal frequency domain displays for LPC Autocorrelation output, although you can override the initialization at any time in the individual display data-entry menus.

To achieve linear predictive coding, Hypersignal offers the autocorrelation method of LPC analysis, in the form of the “LPC Autocorrelation” menu selection. This selection enables very fast, flexible LPC analysis of time domain waveforms. Output coefficients may be either reflection or analysis filter coefficients. Output coefficients are output to a standard Hypersignal time domain file compatible with other time domain functions. Assuming the input waveform was semi-periodic (such as a speech waveform), subsequent FFT processing of these coefficients will produce a highly smoothed frequency response of the original input time waveform.

Autocorrelation files subsequently transformed into the frequency domain carry a parameter that directs the magnitude and spectrogram display functions to automatically depict waveform data as negative log, unless explicitly overridden by the user.

LPC Autocorrelation Menu Fields

The following paragraphs explain the data-entry menu fields for the LPC Autocorrelation function:

INPUT FILENAME:

This field should specify the name of the input time waveform file to be analyzed.

The filename specified should be located on the current drive and subdirectory displayed in the user setup area. The current logged drive and subdirectory can be changed at any time through use of the User Setup function under the UTILITIES column heading on the main menu (see section 2.5.1, Initializing the User Setup Area). If the file does not exist on the current drive and subdirectory, a directory of “.TIM” files located on the current drive and subdirectory will appear in the System Status area. Use the ‘PgDn’ key to page down in the pop-up directory display, and the ‘PgUp’ key to return to the first page of the directory. The search strategy for pop-up directory display will change with consecutive use of the ‘PgUp’ key—see section 3.4.2, Pop-Up Directory Display.

OUTPUT FILENAME:

This field should specify the name of an output time file to contain the filter coefficients.

FRAMESIZE:

This field should specify the framesize to use in segmenting the input time domain waveform. Limits are:

$$(\text{LPC ORDER}+1) \leq \text{FRAMESIZE} \leq 4096$$

See section 4.4.1, Framesize and Overlap Discussion for greater detail on choosing appropriate values of framesize.

Initially FRAMESIZE is read from the currently selected input waveform file header (if not zero) but may be changed at any time. Note that each time the INPUT FILENAME field is entered, the framesize will be re-initialized. The framesize stored in the waveform file header is never modified by this function.

NOTE: **When producing autocorrelated time domain waveform output for further FFT processing, note that the FRAMESIZE field has a counterpart in the FFT Generation data-entry menu; but, if specified here, then the FFT Generation framesize should be set to LPC ORDER+1 (which is how it will be initialized by the program). FFT processing performed in this way will allow exactly time-synchronized comparison of autocorrelated output and unautocorrelated input in the frequency domain if the FFT Generation on the *unautocorrelated input* uses exactly the same framesize and overlap parameters as used in this function to generate the autocorrelated output.**

LPC ORDER:

This field should specify the number of analysis filter or reflection coefficients. The maximum filter order that may be selected is 31. Also, LPC ORDER may not exceed FRAMESIZE-1 (see FRAMESIZE field, above).

The higher the order, the better the output filter response approximation to the original waveform.

NOTE: **LPC ORDER+1 is the framesize value stored in the waveform header and used for subsequent FFT Generation processing. See section 4.4.1, Framesize and Overlap Discussion, for more information on FFT Generation of LPC Autocorrelated output.**

% FRAMESIZE OVERLAP:

Amount of overlap between successive frames of the input time waveform. Overlap can be pictured as samples at the end of one time frame and the beginning of the next that are in common, or used more than once, when the frames are input to the FFT function. Minimum overlap is zero, in which case no samples are in common between successive time frames; maximum overlap is one less than FRAMESIZE, in which case all samples but one are used more than once.

Essentially, use of overlap increases the number of FFT frames output for a given amount of time, producing increased "time resolution." This can be useful when displaying the results using the 3-D Spectrogram or 2-D Spectrograph functions, which depict frequency and amplitude versus time.

See section 4.4.1 below, Framesize and Overlap Discussion, for greater detail on choosing appropriate FRAMESIZE and OVERLAP values.

NOTE: **When producing autocorrelated time domain waveform output for further FFT processing, note that the % FRAMESIZE OVERLAP field has a counterpart in the FFT Generation data-entry menu; but, if specified here, then the FFT Generation overlap should be zero (which is how it will be initialized by the program). FFT processing performed in this way will allow exactly time-synchronized comparison of autocorrelated output and unautocorrelated input in the frequency domain if the FFT Generation on the *unautocorrelated input* uses exactly the same framesize and overlap parameters as used in this function to generate the autocorrelated output.**

WINDOW:

This field specifies the type of window to be applied to each input frame of the time waveform prior to being processed. Possible entries are:

'R'	rectangular (i.e. none)
'Hm'	Hamming
'Hn'	Hanning
'Bl'	Blackman
'Bt'	Bartlett

Windows are always calculated as being FRAMESIZE samples long and are applied to each frame of the input data symmetrically. Also note that when FRAMESIZE is even, the actual "center point" of the window lies on a "half-sample", causing the points FRAMESIZE/2 and FRAMESIZE/2+1 of the window to be equal.

NOTE: **When producing autocorrelated time domain waveform output for further FFT processing, note that the WINDOW field has a counterpart in the FFT Generation data-entry menu; but, if specified here, then the FFT Generation window should be rectangular (which is how it will be initialized by the program). FFT processing performed in this way will allow exact amplitude comparison of autocorrelated output and unautocorrelated input in the frequency domain if the FFT Generation on the *unautocorrelated input* uses exactly the same framesize, overlap, and window parameters as used in this function to generate the autocorrelated output.**

OUTPUT COEFFICIENTS:

This field should specify the type of output coefficients generated. Possible entries are:

'A'	causes output coefficients to be "analysis filter" coefficients, which can be used create the "synthesis" filter needed to reconstruct an approximation to the input waveform
'R'	causes the output coefficients to be "reflection" coefficients, derived as intermediate results during the linear-predictive analysis

LPC Autocorrelation Example

The macro file "LPCAUTO.MAC" on distribution diskette #2 contains an example of LPC Autocorrelation analysis, and frequency domain before-and-after comparison. See section 3.10, Invoking Macros, for instructions on how to execute macro files. Also, you can use a text-editor to examine the macro file closely. Below is the data-entry menu sequence used for this example, starting with LPC Autocorrelation of the waveform file "speech" provided on distribution diskette #3:

INPUT FILENAME:	speech
OUTPUT FILENAME:	sp_auto
FRAMESIZE:	128
LPC ORDER:	15
% FRAMESIZE OVERLAP:	50
WINDOW (R/Hm/Hn/Bl/Bt):	hm
OUTPUT COEFFICIENTS (A/R):	A

Next, after the LPC Autocorrelation, FFT analyis of the resulting filter coefficients can be performed. Note that FRAMESIZE is set to LPC ORDER+1, and the use of rectangular window and no overlap, since these parameters were already specified in the LPC Autocorrelation function:

FILENAME:	sp_auto
FRAMESIZE:	16
FFT ORDER:	9
% FRAMESIZE OVERLAP:	0
WINDOW (R/Hm/Hn/Bl/Bt):	R
REAL/IMAG OUTPUT (Y/N):	N
PRECISION (I/R):	R
PHASE/MAG APPROX (Y/N):	N

As a next step, FFT processing is also performed on the original, un-autocorrelated input, in order to ensure idential frequency domain processing for later comparison. Note that the FRAMESIZE, % OVERLAP, and WINDOW parameters are idential to those used in the LPC Autocorrelation menu:

FILENAME:	speech
FRAMESIZE:	128
FFT ORDER:	9
% FRAMESIZE OVERLAP:	50
WINDOW (R/Hm/Hn/Bl/Bt):	hm
REAL/IMAG OUTPUT (Y/N):	N
PRECISION (I/R):	R
PHASE/MAG APPROX (Y/N):	N

Finally, as a last step, the results are dual-displayed in the 2-D Spectrograph frequency domain display function, with the LPC Autocorrelated output on the top trace:

FILENAME1:	sp_auto
FILENAME2:	speech
POWER SPEC (P/M):	M
AMP SQUARE (Y/N):	N

```
LOG (N/+/-,N/base^NN,min):   -,N
AMP SCALING (TOP,BOT,REF):   0,0,A
COLOR (Cn/Gn/M):             C2
SURFACE (C/R):               C
TIME SHOWN (S, 0=ALL):       1.0
```

4.4.5 Inverse FFT Generation

The Inverse FFT Generation function allows an inverse Fast Fourier Transform to be performed on a complex frequency domain waveform. Input muse be real and imaginary waveform files (".REA" and ".IMA" filename extensions).

Normally complex waveform files are created from a previous FFT Generation function *performed with the* REAL/IMAG OUTPUT *field enabled* or the Transfer Function menu selection, but can also be generated outside of Hypersignal (see section 3.8.3, Creating Frequency Domain Waveform Files). Output is the time domain waveform corresponding to the complex input data, and may be complex or "real", depending on the COMPLEX ATTRIBUTE in the waveform file header (see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT).

Inverse FFT Generation Menu Fields

The following sections explain the Inverse FFT Generation menu data-entry fields:

FILENAME:

This field specifies the name of the file containing the input complex frequency domain waveform. If not found, a directory of ".REA" or ".IMA" files will appear in the System Status area; use the 'PgUp' and 'PgDn' keys to search for a waveform (see section 3.4.2, Pop-Up Directory Display).

PRECISION:

> **NOTE:** **This field is not active (does not appear on the menu) if a DSP-based hardware is currently selected in the DSP ACCELERATOR field of the System Configuration menu.**

When DSP hardware is used to perform the inverse FFT processing, the precision used is the natural word-length of the DSP chip. For example, fixed-point DSP hardware might use 16-bit or 24-bit precision. Floating-point DSPs normally perform all inverse FFT processing using 32-bit floating-point calculations. However, if the DSP ACCELERATOR field is set to 'none', then the PRECISION field is active and the following entries are possible:

'R'	Specifies 8-byte floating-point precision (unless inverse FFT size is 4096, in which case 4-byte floating-point precision is used).
'I'	Specifies integer precision (16x16 multiply, 32-bit add/subtract, and 32x16 divide/shift). Integer precision invokes 80x86 assembly language routines, and provides greatest speed, but with greater chance of numerical overflow. If overflow is detected, the program will automatically rescale the input data by 0.5 and re-start, repeating this process as necessary to achieve no overflow. The actual input waveform file is never modified. Overflow using real precision can occur only if display results cannot be stored with integer precision, in which case scaling will again be automatic, repeating as necessary.

Inverse FFT Generation Notes

If an AMPLITUDE EXPONENT scaling factor is present inside the complex waveform file headers (resulting from a previous forward FFT operation), it is accounted for during inverse FFT processing. Also, overlap and framesize values found in the waveform header are likewise applied to re-create the correct time domain sample sequence.

It is strongly suggested that precision not be mixed between forward and subsequent inverse FFT operations—especially integer forward FFT and real inverse FFT, which may produce a substantial loss of accuracy in reproducing the original time do-

main waveform. Thus, it is suggested that when future inverse FFT operations are anticipated, real precision be used if possible. For most operations, use of a floating-point DSP accelerator qualifies as “real precision”.

4.5 Displaying Frequency Domain Waveforms

There are many ways to display frequency domain waveforms in Hypersignal. Both frame-by-frame and multiple-frame displays are available; within these displays are standard interactive options such as zoom, pan, random-access, scaling, and print-out. Multiple-frame displays include a “waterfall” type display (3-D Spectrogram) and a “contour” type display, where you are “looking down” at the data (2-D Spectrograph). Due to the “three-dimensional” nature of these type of displays (frequency, amplitude, and time), their interactive options include such non-standard choices as “dB/step”, “topology”, “apparent angle of incline”, and “hidden-line removal”.

Please note that there are several examples of frequency domain waveform display in the Tutorial, including step-by-step menu set-ups and interactive display commands.

4.5.1 2-D Spectrograph Display

The Hypersignal 2-D Spectrograph may be the most useful frequency domain display that you will use. Among frequency domain displays, it allows the greatest amount of data to be considered at one time, because amplitude information is encoded as color or gray-level. This avoids the need for a “landscape” topology which might hide some parts of the data, and/or require extra display space to create the illusion of a 3-dimensional surface.

In a Hypersignal 2-D Spectrograph display, multiple FFT-output frames are displayed at once, with time along the horizontal axis, frequency along the vertical, and amplitude shown as the flat surface at which you are “looking down”. You control the number of FFT-output frames (equivalent to an amount of time) shown in each display segment through a “time shown” data-entry field, which takes entry in seconds. If, because a large amount of time has been specified for viewing at once, several FFT-output frames map to the same vertical pixel line on the screen, the maximum of amplitude values occuring at different frequency points for these frames is actually plotted.

To increase the frequency resolution (vertical resolution) of 2-D Spectrograph displays, use a larger FFT length in the FFT Generation function. To increase the time resolution (horizontal resolution) of 2-D Spectrograph displays, use a larger amount of overlap in the FFT Generation function. Especially when performing a zoom operation in the display, increased time resolution may be very helpful. Overall, increased time resolution can markedly improve the clarity of most 2-D Spectrograph displays. See section 4.4.1, Framesize and Overlap Discussion, for suggestions on how to control time resolution.

In some instances, people may refer to “wideband” and “narrowband” analysis when using a 2-D Spectrograph display. To achieve wideband analysis, increase the framesize parameter in the FFT Generation function; decrease for narrowband analysis.

2-D Spectrograph Menu Fields

The following paragraphs explain the 2-D Spectrograph data-entry menu fields:

FILENAME1:

Specifies the filename of a frequency domain magnitude or power spectra waveform to be displayed. The filename specified should be located on the current drive and subdirectory displayed in the User Setup area. The current drive and subdirectory can be changed at any time through use of the User Setup function under the Utilities column heading on the main menu (see section 2.5.1, Initializing the User Setup Area).

If the waveform file does not exist on the current drive and subdirectory, an audible warning will be issued, the cursor repositioned to the filename field, and a directory of “.FRQ” files will be displayed. Paging through the directory is accomplished by using the ‘PgDn’ key to scroll down, and the ‘PgUp’ key to return to the first page of the directory. Consecutive use of the ‘PgUp’ key will change the search strategy for pop-up directory display—see section 3.4.2, Pop-Up Directory Display, for details.

FILENAME2:

This field specifies the filename of a waveform to be dual-displayed with FILENAME1, above. If this field is left blank (you can use 'Delete' or 'Backspace' keys, or press 'Enter' with the cursor at the start of the field to blank the field), only FILENAME1 will be displayed.

POWER SPECTRA:

The POWER SPECTRA field determines whether the input frequency domain waveform is magnitude (".FRQ" filename extension) or power spectra (".PWR" filename extension). Possible entry can be:

'P'	enables power spectra display for the waveform(s)
'M'	enables standard magnitude display (default)

NOTE: **The Power Spectra Gen menu selection *must have been executed previously on the magnitude waveform in order to create the ".PWR" power spectra file required by the 'P' option.***

All display options and interactive commands function with power spectra waveform files exactly as with magnitude waveform files.

AMP SQUARE:

A 'Y' entry enables amplitude-squaring of waveform display values. Amplitude-squared is the technically correct method of power spectra display, but not required by any means. Default entry is 'N', which disables amplitude-squared display.

LOG:

This field controls whether amplitude (color or gray-level encoded) and frequency (y-axis) are shown in linear or log formats. Entry should be in the form of:

amp,base^C,min

where "amp" controls whether amplitude is linear, log, or negative log, "base" is non-zero to set the number base of the logarithm that will be used for the frequency axis, "C" the number of cycles that will be visible if the frequency axis is log, and "min" the minimum cutoff value if the frequency axis is log (in Hz).

The Hypersignal default entry is '+', which specifies log amplitude and linear frequency. See section 4.5.3.1, Log vs. Linear Frequency Domain Display, for examples of possible entry.

AMP SCALING:

This field specifies the type of amplitude scaling for both linear and log amplitude displays. Entry is of the form:

TOP,BOT,REF

For linear and log amplitude displays, TOP and BOT set the vertical plot limits. For linear displays, TOP and BOT should be entered in the linear magnitude units (see section 4.5.3.1 below, Linear vs. Log Frequency Domain Display); for log displays TOP and BOT should be entered in dB.

For log amplitude display only, the REF entry is used to select the mode for dB "reference value". REF entry can be as follows:

'A'	autoscale, which causes the maximum peak in the waveform to be defined as 0 dB
'S'	enables the "system reference" mode of display, in which an outside "system reference value" is considered to be 0 dB; the system reference value is set such that filter response and transfer function waveforms will display with unity gain as 0 dB
'C'	enables the "system calibration" mode of display, which is similar to the system reference mode described above, except offset by the current calibration value set in the Spectrum Analyzer (assigned using engineering dB units prompt)
'W'	enables the "waveform calibration" mode of display, which is similar to the system reference mode described above, except offset by the current calibration offset contained in the waveform header (stored there as the current Spectrum Analyzer calibration offset at the time the waveform was acquired)

The system default entry for this field is '0,0,A', which causes autoscale display of the maximum peak of FILENAME1 and FILENAME2 waveforms as 0 dB. The system reference value, calibration offset, and dB engineering units are not used in linear amplitude display.

Regardless of log or linear amplitude display, if you wish to make both waveforms appear to have the same peak, set the DUAL AUTOSCALE field in the User Setup menu to 'ON'.

See section 4.5.3.1 below, Linear vs. Log Frequency Domain Display, for greater detail on how to use the REF entry for this field.

Note that in the 2-D Spectrograph function, the "step", or quantization value for each color or gray level is equal to (TOP-BOT)/8. This value is shown at the upper left during display. Waveform values less than BOT will not be visible (will show as "9th level", or black). When using the default entry of '0,0,A', a step value of 5 dB per quantization level is preset (i.e. the same display could be achieved with an entry of '0,-40,A'). If you need to see a specific dB value N per quantization level, your entry should be of the form '0,-8*N,A', '0,-8*N,S', or '0,-8*N,W', depending on the type of peak reference desired.

COLOR:

Possible entries for this field are:

'C1'	Enables an 8-level "rainbow" color-encoding scheme, which maps colors to amplitude as they occur in the natural visible-light spectrum.
'C2'	Enables an 8-level "heat" color-encoding scheme, which maps colors to amplitude as a piece of steel would appear as it was heated to be "white-hot". See the distribution disk file "CONFIG.DOC" for details on changing the default color schemes.
'G1'	Enables a "gray-level" display which will function correctly only with VGA (and "SuperVGA") and PS/2 compatible video hardware that supports the "gray-level summing" command present in versions of the ROM BIOS that contain a "VGA BIOS". The video hardware must utilize an "analog video DAC", and be set up for and and connected to an analog monitor in order to allow this selection to work correctly.
'M'	Enables monochromatic display (two-colors only), which, when used in conjunction with a raster surface option (below), will create a simulated gray-level display by varying the density of "on" pixels. However, monochromatic display works well only when data has been spread out (smaller TIME SHOWN value, or zoom mode—see below), a definite disadvantage compared to color or true gray-level displays.

NOTE: When performing a publication-quality print (use 'F2' key) from the 2-D Spectrograph function, remember that EMS memory requirements are the same as for a 16-color printer, because the print function will try to achieve "simulated" gray-scale operation with black & white printers. This is done with a technique similar to a newspaper comic strip, where the density of dots is used to create the illusion of different shades of gray.

SURFACE:

Possible entries for this field are:

'C'	enables contour display
'R'	enables a sort of "raster" display

Contour display interpolates between amplitude values and maps intermediate levels to color accordingly, in much the same manner as drawing lines between points is done in the 3-D Spectrogram function. Raster display indicates amplitude by thickness, or darkness between successive frequency frames (and also color, if enabled). Contour display is highly effective for color or gray-level display mode, and also for most printers when the 'F2' print function is invoked (see note above). Raster display is best suited for monochrome displays/monitors.

TIME SHOWN:

Entry should be a value specifying the amount of time (in seconds) that will be spanned by each screen of spectrographic display. An integer number of FFT-output frames will be displayed, based on the "effective framesize" for each FFT-output frame,

that is as close as possible to the amount of time requested. Effective framesize is the original FFT Generation framesize minus original overlap.

During display, pan commands can be used to view preceding and following screens, and random-access time-entry can be used to arbitrarily position the start of a display screen. A zero entry will display the entire waveform.

Should multiple FFT-output frames map to the same vertical line of pixels in the display, then the maximum of each frequency point will be displayed. In this manner, data features that create higher amplitude, such as a discontinuity or "spike", will always be visible.

4.5.2 3-D Spectrogram Display

The Hypersignal 3-D Spectrogram display is also referred to as a "landscape" or "waterfall" type of display. Multiple FFT-output frames are displayed at once, with frequency along the horizontal axis, amplitude along the vertical axis, and time along a third, "apparent" axis, created by drawing each FFT-output frame slightly behind the previous frame. You control the number of FFT-output frames (equivalent to an amount of time) shown in each display segment through a "time shown" data-entry field, which takes entry in seconds. If, because a large amount of time is specified for viewing at once, several FFT-output frames map to the same horizontal line in the landscape, the maximum of amplitude values occuring at different frequency points for these frames is actually plotted.

To increase the frequency resolution (horizontal resolution) of 3-D Spectrogram displays, use a larger FFT length in the FFT Generation function. To increase the time (depth, or "apparent", axis) resolution of 3-D Spectrogram displays, use a larger amount of overlap in the FFT Generation function. Overall, increased time resolution can markedly improve the "smoothness" of the landscape in 3-D Spectrogram displays. See section 4.4.1, Framesize and Overlap Discussion, for suggestions on how to control time resolution.

3-D Spectrogram Menu Fields

The following paragraphs explain the data-entry menu fields in the 3-D Spectrogram function:

FILENAME1:

Specifies the filename of a frequency domain magnitude or power spectra waveform to be displayed. The filename specified should be located on the current drive and subdirectory displayed in the User Setup area. The current drive and subdirectory can be changed at any time through use of the User Setup function under the Utilities column heading on the main menu (see section 2.5.1, Initializing the User Setup Area).

If the waveform file does not exist on the current drive and subdirectory, an audible warning will be issued, the cursor repositioned to the filename field, and a directory of ".FRQ" files will be displayed. Paging through the directory is accomplished by using the 'PgDn' key to scroll down, and the 'PgUp' key to return to the first page of the directory. Consecutive use of the 'PgUp' key will change the search strategy for pop-up directory display—see section 3.4.2, Pop-Up Directory Display, for details.

FILENAME2:

This field specifies the filename of a waveform to be dual-displayed with FILENAME1, above. If this field is left blank (you can use 'Delete' or 'Backspace' keys, or press 'Enter' with the cursor at the start of the field to blank the field), only FILENAME1 will be displayed.

POWER SPECTRA:

The POWER SPECTRA field determines whether the input frequency domain waveform is magnitude (".FRQ" filename extension) or power spectra (".PWR" filename extension). Possible entry can be:

'P'	enables power spectra display for the waveform(s)
'M'	enables standard magnitude display (default)

NOTE: **The Power Spectra Gen menu selection *must have been executed previously on the magnitude waveform in order to create the ".PWR" power spectra file required by the 'P' option.***

All display options and interactive commands function with power spectra waveform files exactly as with magnitude waveform files.

AMP SQUARE:

A 'Y' entry enables amplitude-squaring of waveform display values. Amplitude-squared is the technically correct method of power spectra display, but not required by any means. Default entry is 'N', which disables amplitude-squared display.

LOG:

This field controls whether amplitude (y-axis) and frequency (x-axis) are shown in linear or log formats. Entry should be in the form of:

amp,base^C,min

where "amp" controls whether amplitude is linear, log, or negative log, "base" is non-zero to set the number base of the logarithm that will be used for the frequency axis, "C" the number of cycles that will be visible if the frequency axis is log, and "min" the minimum cutoff value if the frequency axis is log (in Hz).

The Hypersignal default entry is '+', which specifies log amplitude and linear frequency. See section 4.5.3.1, Log vs. Linear Frequency Domain Display, for examples of possible entry.

SURFACE,GRID:

This field controls hidden-line removal and "surface mesh" drawing for the display, and also the type of grid underlay for the display. Entry should be in the form:

SURFACE,GRID

where SURFACE entries can be:

'H'	Enables hidden-line removal
'M'	Also enables hidden-line removal, but in addition enables a "mesh" (or cross-hatch) to enhance the illusion of a 3-D surface. Enabling the surface mesh increases display drawing time significantly.
'N'	Disables both hidden-line removal and the mesh

GRID entries can be:

'N'	disables all grid display
'S'	enables solid-line grid display
'I'	enables "intersected" grid display, in which case only the actual points of intersection between horizontal and vertical grid are displayed

The Hypersignal default entry for this field is 'H,N', which specifies hidden-line removal, no mesh, and no grid.

Y-LABELS:

For linear amplitude display only, entry in this field in the form

SCALE FACTOR,UNITS

will cause the y-axis units to be multiplied by SCALE FACTOR (a numeric value) prior to display, and UNITS (a text, or string value) to be written (labelled) vertically along the y-axis. Some entry examples:

'0.123,(Volts)'	scale factor is 0.128, units will be labelled on the vertical axis as "(Volts)"
'1.789,Psi'	scale factor is 1.789, units will be labelled on vertical axis as "Psi"

If you omit the UNITS portion of the entry, no label will be written. If you blank the field entirely, then the scale factor is assumed to be 1. The actual waveform data file is not modified.

AMP SCALING:

This field specifies the type of amplitude scaling for both linear and log amplitude displays. Entry is of the form:

TOP,BOT,REF

For linear and log amplitude displays, TOP and BOT set the vertical plot limits. For linear displays, TOP and BOT should be entered in the current units (see Y-LABELS field, above); for log displays TOP and BOT should be entered in dB.

For log amplitude display only, the REF entry is used to select the mode for dB "reference value". REF entry can be as follows:

'A'	autoscale, which causes the maximum peak in the waveform to be defined as 0 dB
'S'	enables the "system reference" mode of display, in which an outside "system reference value" is considered to be 0 dB; the system reference value is set such that filter response and transfer function waveforms will display with unity gain as 0 dB
'C'	enables the "system calibration" mode of display, which is similar to the system reference mode described above, except offset by the current calibration value set in the Spectrum Analyzer (assigned using engineering dB units prompt)
'W'	enables the "waveform calibration" mode of display, which is similar to the system reference mode described above, except offset by the current calibration offset contained in the waveform header (stored there as the current Spectrum Analyzer calibration offset at the time the waveform was acquired)

The system default entry for this field is '0,0,A', which causes autoscale display of the maximum peak of FILENAME1 and FILENAME2 waveforms as 0 dB. The system reference value, calibration offset, and dB engineering units are not used in linear amplitude display.

Regardless of log or linear amplitude display, if you wish to make both waveforms appear to have the same peak, set the DUAL AUTOSCALE field in the User Setup menu to 'ON'.

See section 4.5.3.1 below, Linear vs. Log Frequency Domain Display, for greater detail on how to use the REF entry for this field.

ANGLE,TIME SHOWN:

ANGLE specifies the "angle of apparent incline" used to create the time dimension in the landscape display. An angle of 90 degrees is a true "waterfall" display, an angle of 5 degrees allows you to view the landscape virtually from the side (default value is 45 degrees, for "isometric" viewing).

TIME SHOWN specifies the amount of frequency domain data that will be displayed in each 3-D Spectrogram screen. An integer number of FFT-output frames will be displayed, based on the "effective framesize" for each FFT-output frame, that is as close as possible to the amount of time requested. Effective framesize is the original FFT Generation framesize minus the overlap. Entry should be in the form:

ANGLE,TIME

where ANGLE should be entered in degrees (from 5 to 90, inclusive), and TIME should be entered in seconds.

During display, pan commands can be used to view preceding and following screens, and random-access time-entry can be used to arbitrarily position the start of a display screen. A zero entry for TIME SHOWN will display the entire waveform.

4.5.3 Magnitude Display

The Magnitude Display menu selection in Hypersignal invokes a frame-by-frame display of frequency domain waveforms with combinations of linear or log amplitude, and linear or log frequency axis. Input to this display function is normally the output of the FFT Generation function. Basically, any time domain waveform can be thought of as the sum of several sinusoidal waveforms of the form:

$S_n(t) = A_n \sin(\Omega_n t + \Phi_n)$

where Ω is the frequency, and Φ the phase. Magnitude display indicates the amplitude A of the possible frequencies Ω contained in the original, pre-FFT time domain waveform, and is calculated as the squareroot of the sum of the squares of the real and imaginary components of FFT output.

The amplitude of magnitude values, as they are stored in Hypersignal frequency domain waveforms, have a direct relationship to the original time domain waveform that was used as FFT input. See section 4.5.3.2 below, Frequency Domain Display Scaling, for details about magnitude amplitude scaling.

Magnitude Display Menu Fields

The paragraphs below explain the data-entry fields in the Magnitude Display function:

FILENAME1:

Specifies the filename of a frequency domain waveform for which a magnitude or power spectra response will be displayed. The filename specified should be located on the current drive and subdirectory displayed in the User Setup area. The current drive and subdirectory can be changed at any time through use of the User Setup function under the Utilities column heading on the main menu (see section 2.5.1, Initializing the User Setup Area).

If the waveform file does not exist on the current drive and subdirectory, an audible warning will be issued, the cursor repositioned to the filename field, and a directory of ".FRQ" files will be displayed. Paging through the directory is accomplished by using the 'PgDn' key to scroll down, and the 'PgUp' key to return to the first page of the directory. Consecutive use of the 'PgUp' key will change the search strategy for pop-up directory display—see section 3.4.2, Pop-Up Directory Display, for details.

FILENAME2:

This field specifies the filename of a waveform to be dual-displayed with FILENAME1, above. If this field is left blank (you can use 'Delete' or 'Backspace' keys, or press 'Enter' with the cursor at the start of the field to blank the field), only FILENAME1 will be displayed.

POWER SPECTRA:

The POWER SPECTRA field determines whether the input frequency domain waveform is magnitude (".FRQ" filename extension) or power spectra (".PWR" filename extension). Possible entry can be:

'P'	enables power spectra display for the waveform(s)
'M'	enables standard magnitude display (default)

NOTE: **The Power Spectra Gen menu selection *must have been executed previously on the magnitude waveform in order to create the ".PWR" power spectra file required by the 'P' option.***

All display options and interactive commands function with power spectra waveform files exactly as with magnitude waveform files.

AMP SQUARE:

A 'Y' entry enables amplitude-squaring of waveform display values. Amplitude-squared is the technically correct method of power spectra display, but not required by any means. The default entry is 'N', which disables amplitude-squared display.

LOG:

This field controls whether amplitude and frequency (y-axis and x-axis) are shown in linear or log formats. Entry should be in the form of:

amp,base^C,min

where "amp" controls whether the vertical axis (amplitude) is linear, log, or negative log, "base" is non-zero to set the number base of the logarithm that will be used for the frequency axis, "C" the number of cycles that will be visible if the frequency axis is log, and "min" the left cutoff value if the frequency axis is log (in Hz).

The Hypersignal default entry is 'N' (none), which specifies linear amplitude and linear frequency. See section 4.5.3.1, Log vs. Linear Frequency Domain Display, for examples of possible entry.

Y-LABELS:

For linear amplitude display only, entry in this field in the form

SCALE FACTOR,UNITS

will cause the y-axis units to be multiplied by SCALE FACTOR (a numeric value) prior to display, and UNITS (a text, or string value) to be written (labelled) vertically along the y-axis. Some entry examples:

'0.123,(Volts)'	scale factor is 0.128, units will be labelled on the vertical axis as "(Volts)"
'1.789,Psi'	scale factor is 1.789, units will be labelled on vertical axis as "Psi"

If you omit the UNITS portion of the entry, no label will be written. If you blank the field entirely, then the scale factor is assumed to be 1. The actual waveform data file is not modified.

AMP SCALING:

This field specifies the type of amplitude scaling for both linear and log amplitude displays. Entry is of the form:

TOP,BOT,REF

For linear and log amplitude displays, TOP and BOT set the vertical plot limits. For linear displays, TOP and BOT should be entered in the current units (see Y-LABELS field); for log displays TOP and BOT should be entered in dB.

For log amplitude display only, the REF entry is used to select the mode for dB "reference value". REF entry can be as follows:

'A'	autoscale, which causes the maximum peak in the waveform to be defined as 0 dB
'S'	enables the "system reference" mode of display, in which an outside "system reference value" is considered to be 0 dB; the system reference value is set such that filter response and transfer function waveforms will display with unity gain as 0 dB
'C'	enables the "system calibration" mode of display, which is similar to the system reference mode described above, except offset by the current calibration value set in the Spectrum Analyzer (assigned using engineering dB units prompt)
'W'	enables the "waveform calibration" mode of display, which is similar to the system reference mode described above, except offset by the current calibration offset contained in the waveform header (stored there as the current Spectrum Analyzer calibration offset at the time the waveform was acquired)

The system default entry for this field is '0,0,A', which causes autoscale display of the maximum peak of FILENAME1 and FILENAME2 waveforms as 0 dB. The system reference value, calibration offset, and dB engineering units are not used in linear amplitude display.

Regardless of log or linear amplitude display, if you wish to make both waveforms appear to have the same peak, set the DUAL AUTOSCALE field in the User Setup menu to 'ON'.

See section 4.5.3.1 below, Linear vs. Log Frequency Domain Display, for greater detail on how to use the REF entry for this field.

FORMAT,PLOT:

This field specifies the format of dual-trace display and also the plot type for data sample depiction. Below are possible dual-trace format entries:

'D'	dual-trace display
'S'	subtracted trace display (assumes two-trace display)
'O'	overlaid-trace display (assumes two-trace display)

Below are possible plot-type entries:

'D'	discrete
'C'	connected
'B'	both (connected and discrete)

All combinations of the above entries are valid. Of course, the FORMAT portion of this field is only active for dual-waveform display (both the FILENAME1 and FILENAME2 fields, above, have been entered).

The connected plot type utilizes linear interpolation between data samples, which might look "continuous", but is not intended to be as precise as higher interpolation orders or lowpass filter reconstruction, and is thus referred to only as "connected". The discrete plot type is the most precise, showing only actual individual sample points, but may not be as readable for large amounts of displayed data. Some examples of entry:

'D,C'	dual-trace, connected display
'O,D'	overlaid, discrete display

GRID:

This field specifies the type of grid underlay for waveform display. Possible entries are:

'N'	turns off all grid display
'S'	enables solid-line grid display
'D'	enables dashed-line grid display
'I'	enables intersection grid display

Grid lines are drawn as extensions of the tic marks on the vertical axis and the major tic marks on the horizontal axis. In the case of intersection grid display, only the actual points where horizontal and vertical grid lines would intersect are displayed. Grid underlay is intended as another aide in measuring waveform magnitude or power spectra values.

4.5.3.1 Linear vs. Log Frequency Domain Display

This section describes three areas:

1) possible entry in the LOG or LOG FREQ fields of the Magnitude Display, Phase Display, 3-D Spectrogram, 2-D Spectrograph, and Spectrum Analyzer functions

2) log amplitude magnitude display, and possible REF entry (and its effect) in the AMP SCALING field of the Magnitude Display, 3-D Spectrogram, and 2-D Spectrograph functions (the REF field only pertains to log amplitude magnitude display)

3) linear amplitude magnitude display

Additional information concerning the relationship of frequency domain linear amplitude units to the original time domain units, which is of vital importance in understanding what you are looking at when you view a frequency domain magnitude display in Hypersignal, is contained in section 4.5.3.2, Frequency Domain Display Scaling, below.

LOG or LOG FREQ Field Entry

Entry in the Magnitude Display, 3-D Spectrogram, and 2-D Spectrograph fields should be in the form of:

amp,base^C,min

Entry in the Phase Display and Spectrum Analyzer fields should be in the form of:

base^C,min

where "amp" is not included, either because its not relevant (Phase Display) or specified in another manner (Spectrum Analyzer).

If "amp" is part of the entry, it controls whether amplitude display is log, negative log, or linear. "base" is the number base of the logarithm that will be used, "C" the number of cycles that will be visible, and "min" the lefthand cutoff (in Hz) for the display. Some examples of possible entry when "amp" is part of the entry:

'n'	amplitude is linear, frequency axis is linear
'+'	amplitude is log, frequency axis is linear
'+,10^3'	amplitude is log, frequency axis is $\log_{10}$, 3 cycles visible
'n,2^12,200'	amplitude is linear, frequency axis is $\log_2$, 12 octaves visible, minimum value displayed is 200 Hz

'-,OCT^6,10'	amplitude is negative log, frequency axis is 1/6 octave-band display, minimum value displayed is 10 Hz
'+,OCT^12,30'	amplitude is log, frequency axis is 1/12 octave-band display, minimum value displayed is 30 Hz
'n,DEC^6,50'	amplitude is linear, frequency axis is 1/6 decade-band display, minimum value displayed is 50 Hz

Some examples of possible entry when "amp" is not part of the entry:

'N'	frequency axis is linear
'10^3'	frequency axis is $\log_{10}$, 3 cycles visible
'2^12,200'	frequency axis is $\log_2$, 12 octaves visible, minimum value displayed is 200 Hz
'OCT^6,10'	frequency axis is 1/6 octave-band display, minimum value displayed is 10 Hz
'OCT^12,30'	frequency axis is 1/12 octave-band display, minimum value displayed is 30 Hz
'DEC^6,50'	frequency axis is 1/6 decade-band display, minimum value displayed is 50 Hz

It should be pointed out that 'OCT' and 'DEC' are merely special cases of 2^N display and 10^N display, respectively, where N is preset to a fixed value (11 in the case of octave band display). The advantage to using this form of entry is that you can additionally specify equally-spaced "subdivisions" of each band that are summed and divided by the number of points in the subdivision.

REF Field Entry (log display only)

To simply "autoscale", or use the maximum peak in the frequency domain waveform as 0 (zero) dB, select the 'A' option. This is the Hypersignal default, and offers the convenience of being able to quickly and easily see log amplitude displays without concern as to relative amplitude between waveforms, system reference value, or calibration offset.

If you wish to compare several waveforms acquired under similar conditions all against the same log scale, without the current calibration offset set in the Spectrum Analyzer function, select the system reference mode ('S' entry in the REF subfield).

If you wish to compare several waveforms acquired under similar conditions all against the same log scale, including the current calibration offset set in the Spectrum Analyzer function, select the system calibration mode ('C' entry in the REF subfield; see section 4.7.2.10, Spectrum Analyzer Amplitude Display and Calibration).

If you want to compare waveforms acquired with different hardware, or under different conditions, use the waveform calibration mode ('W' entry in the REF subfield); this will cause the Spectrum Analyzer calibration settings that active and were stored in the waveforms' headers at the time of waveform acquisition to be applied independently.

The system calibration offset (usually derived from a reference tone peak), and the dB engineering units assigned to it, can be set either in the Spectrum Analyzer function via the "calibration" command (see section 4.7.2.10, Spectrum Analyzer Amplitude Display and Calibration) or directly in the configuration file (see the CONFIG.DOC file for location). The program default is 0 dB, assigned to a value of 32767 (16-bit A/D full-scale). Thus, by defining the peak (or other reference point) of an analog input signal in the Spectrum Analyzer function as an arbitrary dB value, all other frequency domain displays can be viewed with the same scale and units using the system calibration REF setting ('C' entry in the REF subfield).

For log display, entry in the TOP and BOT subfields controls the vertical plot limits in dB, regardless of the REF mode.

Linear Amplitude Display

Linear magnitude amplitude display is set by entering 'N' at the first of the LOG field.

For linear magnitude display, entry in the TOP and BOT subfields control the vertical plot limits in linear units (default is Volts). If the BOT subfield is set to zero, then autoscaling is in effect. If not, then the current value is multiplied by the scale factor entered in the Y-LABELS field, and used to control the vertical plot limit. The REF subfield does not apply to linear magnitude amplitude display.

4.5.3.2 Frequency Domain Display Scaling

In all Hypersignal frequency domain magnitude waveform headers, an AMPLITUDE EXPONENT is stored, which in combination with the MAX AMPLITUDE header value, forms the actual maximum absolute value for the magnitude data. The AM-

PLITUDE EXPONENT value is stored by the FFT Generation function; when used as a power of two it forms a scaling factor for frequency domain data (see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT).

In the Spectrum Analyzer interactive display (real-time software only), the FFT analysis is being performed in real-time by the currently installed DSP hardware. In this case, FFT results are based on 16-bit A/D converter input data and no scale factor is removed from the data.

When magnitude data is displayed in linear form (or when being calculated prior to display in log form), or magnitude data is being processed by a DSP chip prior to uploading results for display in the Spectrum Analyzer, each magnitude point undergoes the following scaling transformation:

1) For waveform file data only, multiplied by $2^{\text{AMPLITUDE EXPONENT}}$.

2) Multiplied by 2 if original time domain data was non-complex (frequency domain display is showing a "single-sided" spectrum); otherwise multiplied by 1 (complex time domain data; double-sided spectrum).

3) Divided by original time domain FRAMESIZE value used in FFT. Note that this is not done if the data represents the frequency transform of a filter impulse response, allowing the gain of the filter to be displayed relative to 0 dB (or unity gain).

4) Divided by a "window factor" (W) to compensate for any time domain window that was applied (can be calculated by adding up all window values, dividing by FRAMESIZE); W = 1 for a rectangular window, W < 1 for all other windows.

5) If, and only if, input represents a "simulated quantized impulse response" produced by the Code Generate/Quantize function, further scaling will be performed to compensate for the quantization and input level values specified during the simulation. This allows filter frequency responses to be compared easily between non-quantized and quantized versions. See section 4.9.5.1, Simulating the Effects of Quantization, for greater detail.

This scaling transformation returns the amplitude units of the magnitude data to same as the original time domain data; for example, Volts. Thus, a pure sinusoid of peak amplitude A in the time domain will also show amplitude A at its fundamental frequency in a linear magnitude display.

Double-Sided Spectrum Display

Scaling for double-sided spectrum displays varies from single-sided displays only in step 2) of the sequence described above. If you need double-sided spectrums, or non-zero imaginary input (sometimes referred to as "analytic sampling") then the following options are available:

1) For waveform file data, use complex time domain waveforms (see section 3.8.1, Creating Complex Waveform Files).

2) For Spectrum Analyzer display, you can directly modify the appropriate "DSP Source Code Interface" (see section 4.7, Using DSP/Acquisition Hardware with Hypersignal).

Spectrum Analyzer Amplitude Scaling

Essentially, scaling is as described in the sequence above. However, some differences exist because of the nature of the Spectrum Analyzer as a practical measurement tool. Most notably:

1) No exponent is applied for analog input data, since, by default, the A/D converters on all hardware supported by Hypersignal are assumed to produce values ranging from -32768 to 32767.

2) Transfer function display is treated like the frequency transform of a filter impulse response—no division by framesize is performed, and no multiplication by two for single-sided spectrum is performed.

3) Transfer function *linear* display is arbitrarily multipled by 32767 so that results will be viewable.

Furthermore, some scaling implementation differences can exist during Spectrum Analyzer operation, especially if a fixed-point DSP chip is in use. For information, a list of the type and sequence of scaling performed by the Spectrum Analyzer function for various DSP chips is given below:

1) 24-bit fixed-point DSPs: output FFT data is scaled by 2/FRAMESIZE/W (see sequence described above) for linear magnitude display in order to show Volts, but for log display better dynamic range is enabled by subtracting LOG_{10}(2/FRAMESIZE/W) *after* the log function has been applied.

2) 16-bit fixed-point DSPs: input time data is scaled by 2/FRAMESIZE/W (see above) before FFTs are performed, to allow FFT results to be stored as 16-bit values. Linear or log magnitude processing and display then takes place.

3) 32-bit floating-point DSPs (includes analog-only hardware that use a DSP coprocessor board): output FFT data is scaled by 2/FRAMESIZE/W (see sequence described above). Linear or log magnitude processing/display is then performed.

4) File input is treated exactly as with other Hypersignal frequency domain displays; see the scaling sequence described in the first part of this section.

4.5.4 Wrapped, Unwrapped, and Group Delay Phase Display

The Phase Display menu selection in Hypersignal invokes a frame-by-frame display of frequency domain waveforms with combinations of wrapped phase, unwrapped phase, and group delay display. Input to this display function is normally the output of the FFT Generation function. Basically, any time domain waveform can be thought of as the sum of several sinusoidal waveforms of the form:

$S_n(t) = A_n\sin(\Omega_n t + \Phi_n)$

where Ω is the frequency, and Φ the phase. Just as in magnitude display, which indicates the amplitude A of the possible frequencies Ω contained in the original, pre-FFT time domain waveform, phase display indicates the phase Φ of individual frequency components.

Wrapped phase refers to the output of a standard four-quadrant ARCTAN(imag/real) function that ranges between $-\pi$ and π. The term "wrapped" refers to the fact that the continuous phase, which might easily reach values with magnitude larger than π, can only be represented in modulo fashion by the ARCTAN output. This results in output phase values that appear to "wrap around" as they get near to π or $-\pi$.

The Phase Display function contains a display option that attempts to "unwrap" standard FFT output, in order to yield a more useful picture of the phase response of a waveform. Phase unwrapping refers to the process of part mathematics, part educated guesswork that goes into creating a continuous phase function using the four-quadrant ARCTAN output. The idea is to see how the phase of the waveform *progressed with frequency*, which requires that phase values be tracked continuously as, for example, they reach π and suddenly become $-\pi$. This type of phase display is similar to the phase portion of a Bode plot.

Continuous phase tracking can become extremely difficult if the data is noisy, with small magnitudes that cause the phase to "jitter" between quadrants of the S-plane unit circle (the boundaries of which are possible "wrap-points"). Another factor which can make phase unwrapping difficult is a lack of frequency resolution (small FFT length), which can create substantial ambiguity in determining when a phase response has actually wrapped and when it legitimately just jumped a large amount.

Also, the Phase Display function contains a group delay display option. Group delay is the negative derivative (slope) of the unwrapped phase with respect to frequency, and indicates the delay that would be incurred at different frequencies if the waveform were to be used as a filter (were to represent the transfer function of some system). For example, if several sinusoidal waveforms were sent down the phone line at the same time, some would be delayed more than others, because the phone line does not have linear phase (the slope of the phase is not a constant). As another example, the value of group delay (displayed in msec) for a FIR filter that has linear phase should be equal to half the FIR filter's time domain impulse response length, which is the maximum length any frequency can be delayed through an FIR filter.

Phase Display Menu Fields

The paragraphs below explain the data-entry fields of the Phase Display function:

FILENAME1:

Specifies the filename of a frequency domain waveform for which a phase response will be displayed. The filename specified should be located on the current drive and subdirectory displayed in the User Setup area. The current drive and subdirectory can be changed at any time through use of the User Setup function under the Utilities column heading on the main menu (see section 2.5.1, Initializing the User Setup Area).

If the waveform file does not exist on the current drive and subdirectory, an audible warning will be issued, the cursor repositioned to the filename field, and a directory of ".PHZ" files will be displayed. Paging through the directory is accomplished by

using the 'PgDn' key to scroll down, and the 'PgUp' key to return to the first page of the directory. Consecutive use of the 'PgUp' key will change the search strategy for pop-up directory display—see section 3.4.2, Pop-Up Directory Display, for details.

FILENAME2:

This field specifies the filename of a waveform to be dual-displayed with FILENAME1, above. If this field is left blank (you can use 'Delete' or 'Backspace' keys, or press 'Enter' with the cursor at the start of the field to blank the field), only FILENAME1 will be displayed.

DISPLAY1:

This field specifies the type of display format for the waveform entered in the FILENAME1 field. Below are possible entries for this field:

'W'	selects wrapped phase display
'U'	selects unwrapped phase display
'GD'	selects group delay display.

DISPLAY2:

This field allows the same entry as the DISPLAY1 field, but controls the display type for the FILENAME2 field input waveform.

Note that by specifying the same waveform in both the FILENAME1 and FILENAME2 fields, combinations of wrapped phase, unwrapped phase, and group delay can be compared simultaneously.

LOG FREQUENCY:

This field controls the frequency axis (x-axis) for the display. Entry should be in the form of:

base^C,min

where "base" is non-zero to set the number base of the logarithm that will be used for the frequency axis, "C" the number of cycles that will be visible if the frequency axis is log, and "min" the lefthand cutoff value if the frequency axis is log (in Hz).

The Hypersignal default entry is 'N' (none), which specifies a linear frequency axis. See section 4.5.3.1, Log vs. Linear Frequency Domain Display, for examples of possible entry.

GROUP DELAY SMOOTHING:

The GROUP DELAY SMOOTHING field controls the number of points used in a linear regression calculation of the instantaneous slope (negative derivative) of the unwrapped phase. This calculation is performed for every frequency point in the waveform, and the results are used as group delay values. The GROUP DELAY SMOOTHING factor simply specifies the number of unwrapped phase values to use in each linear regression calculation.

Smaller values will produce sharper and more responsive group delay output, but may also enhance noisy phase areas where the corresponding magnitude display may be small, indicating that phase has little significance in these areas. A value in the range of 7-12 is suggested. Minimum entry is 2 and maximum is 50. This field is ignored if neither DISPLAY1 nor DISPLAY2 specifies group delay display.

DEGREES/RADIANS:

This field simply controls whether the vertical-axis scale will be in degrees or radians. Entry for this field should be either:

'D'	display units are in degrees
'R'	display units are in radians

This field has no effect when only group delay display is currently active, since group delay output is always displayed in milliseconds (msec).

FORMAT,PLOT:

This field specifies the format of dual-trace display and also the plot type for data sample depiction. Below are possible dual-trace format entries:

'D'	dual-trace display
'S'	subtracted trace display (assumes two-trace display)
'O'	overlaid-trace display (assumes two-trace display)

Below are possible plot-type entries:

'D'	discrete
'C'	connected
'B'	both (connected and discrete)

All combinations of the above entries are valid. Of course, the FORMAT portion of this field is only active for dual-waveform display (both the FILENAME1 and FILENAME2 fields, above, have been entered).

The connected plot type utilizes linear interpolation between data samples, which might look "continuous", but is not intended to be as precise as higher interpolation orders or lowpass filter reconstruction, and is thus referred to only as "connected". The discrete plot type is the most precise, showing only actual individual sample points, but may not be as readable for large amounts of displayed data. Some examples of entry:

'D,C'	dual-trace, connected display
'O,D'	overlaid, discrete display

GRID:

This field specifies the type of grid underlay for waveform display. Possible entries are:

'N'	turns off all grid display
'S'	enables solid-line grid display
'D'	enables dashed-line grid display
'I'	enables intersection grid display

Grid lines are drawn as extensions of the tic marks on the vertical axis and the major tic marks on the horizontal axis. In the case of intersection grid display, only the actual points where horizontal and vertical grid lines would intersect are displayed. Grid underlay is intended as another aide in measuring waveform phase values.

4.5.4.1 Suggestions for Unwrapped Phase Display

Phase-unwrapping is not a deterministic process. When Hypersignal processes phase waveforms that have been created with arbitrary FFT sizes, amounts of overlap, and time domain windows, there is no way to be mathematically certain that a point that appears to need unwrapping might not in fact be, a legitimate "jump" in the phase response, resulting from noise, or numerical ambiguity due to low FFT resolution or the effects of a time domain window. Thus, please keep in mind the following guidelines when performing FFT processing on waveforms to be analyzed with unwrapped phase and group delay display:

1) Zero-filling (extending the FFT length) produces greater frequency resolution, which often helps the Hypersignal unwrapping algorithm in resolving numerical ambiguity and achieving more certainty about noise.

2) Usually, more overlap will also help the unwrapping algorithm, but not always.

3) Use real precision whenever possible, to eliminate quantization error.

Also, in some cases it is not a good idea to enable unwrapping, especially if the phase waveform obviously does not need it. The unwrapping algorithm's "big picture" view of slope and change-in-slope (second-derivative) may not be sufficient to ignore some large changes in slope which may actually be valid. LPC autocorrelation data (LPC filter coefficients run through an FFT) are a good example of this: such data is usually relatively smooth, except for occasional phase excursions near the major frequency "formants".

4.6 Waveform Generation and Mathematics

Hypersignal contains a very flexible function called “Difference Equations” which accepts standard difference equation text input. In addition to built-in input such as unit-step, delta, noise, and feedback, equations can also accept external input from “x1" and ”x2" waveform sources. This, combined with a wide variety of operators and functions, allows the Difference Equations function to be used for both generating new waveforms and operating on existing waveforms. Below are some Difference Equations highlights:

1) A sample index “n” and a frame index “m” are available, allowing operation with both time and frequency domain waveforms.

2) Input arrays includes unit-step and delta functions u[n] and d[n], external waveforms x1[n] and x2[n], and recursive input (feedback) y[n].

3) A limited imaginary operator “j” is available, allowing creation of complex waveforms.

4) Nomenclature is standard DSP textbook. Input and indexes are indicated by an array or waveform name and brackets, as in “y[n]”. Level of precedence may be controlled with parentheses or brackets. For example, delays and sample shifts are indicated as relative subtraction from an index, as in “x1[n-M]”, where M is an amount of delay.

5) Operations and functions include signal arithmetic (+, -, ∗, /, modulus), trigonometric and transcendental functions, signal calculus (integration and differentiation), and other functions such as a pseudo-random number generator, mimimum and maximum detection, and several built-in constants.

4.6.1 Writing Difference Equations

Some of the important concepts in the Difference Equations function are discussed below in greater detail. Understanding these concepts is essential to writing concise, effective difference equations.

Sample Index

The sample index, “n”, starts at zero and increments either until the end of all input waveforms specified, or until the number of output samples specified by the # OUTPUT SAMPLES field value have been stored. For example, to decimate an input waveform x1 by every other sample, you could write:

$$y[n] = x1[n*2]$$

Multiplying n by 2 causes samples 0, 2, 4, etc. from the x1 waveform to be stored in the output waveform, and the odd samples to be thrown away. To keep only the odd samples, you could write:

$$y[n] = x1[n*2+1]$$

As another example, consider the running average equation for a time domain waveform below:

$$y[n] = y[n-1] + x1[n] - x1[n-Fr]$$

In this example, the current sample value is added to the running sum “y[n]”, and a sample value “Fr” samples previous is subtracted. “Fr” is a pre-defined constant, and is replaced by the value entered in the FRAMESIZE field (see below) when the equation is evaluated. Note that whenever an array index evaluates to less than zero, input is considered to be zero—no input values are defined for negative time. To maintain the same amplitude scale as the x1 input, the above equation can be modified as follows:

$$y[n] = y[n-1] + (x1[n] - x1[n-Fr]) / Fr$$

Frame Index

Use of a frame index, “m”, is also allowed. The frame index starts at zero and increments only when the sample index increments past a frame boundary. For example, to multiply every frame in an input waveform x1 by a window contained in waveform x2, you could write:

$$y[m,n] = x1[m,n] * x2[0,n]$$

where "m" increments *each time "n" increments an amount specified by the* FRAMESIZE *field*. Thus, in the above example, if we say that FRAMESIZE is equal to 100, then when n=1000, m=10. The instantaneous value of any "frame,offset" expression such as the one used for the "y" output index above can be calculated as follows:

frame∗FRAMESIZE + offset MOD FRAMESIZE

where FRAMESIZE is the value of the FRAMESIZE field, below, and "MOD" indicates modulus division. For this reason, the "," (comma) is referred to as the "frame operator" in the list of operators given below.

Note that in the above example, no matter how large "n" becomes, the "0,n" index expression for the x2 waveform input will always evaluate in a range between 0 and FRAMESIZE-1, inclusive.

With the frame index, not only is specific frame access possible, but functions such as convolution and cross-correlation, which require advancing the starting point for each new frame by only one sample per frame, can also be performed. For example, a cross-correlation operation that slides a "pattern" frame along an input waveform in order to find the best "match", could be written as:

y[m] = y[m] + x1[0,n] ∗ x2[0,m]

where x1 represents the pattern data, and x2 the target waveform. The "0,m" expression causes the starting point for the pattern match to always be in frame 0, but increase by 1 every FRAMESIZE points. In this example, the FRAMESIZE field would contain the length of the x1 pattern waveform.

Input Arrays

In the input array list below, "index" refers to any legitimate expression, which can be a combination of "n" (sample index), "m" (frame index), numbers, functions, operators, and even other input arrays. When used as array indexes, expressions are always truncated prior to actually accessing the array. Array index expressions may also be a function of another array access, i.e. x1[x1[n]+2∗n]. Input arrays include the following:

x1[index]	refers to waveform specified by X1 INPUT FILENAME field
x2[index]	refers to waveform specified by X2 INPUT FILENAME field
y[index]	refers to previous output, sometimes called recursive output or feedback
u[index]	refers to the "unit-step" function; accesses to this array return 0 if index is less than zero, and 1 if index >= zero
d[index]	refers to the "delta" function; accesses to this array return 0 if index is <> zero, and 1 if index = zero
x1'[index] or dx1[index]	refers to derivative of waveform specified by X1 INPUT FILENAME field
x2'[index] or dx2[index]	refers to derivative of waveform specified by X2 INPUT FILENAME field
y'[index] or dy[index]	refers to derivative of previous output
ix1[index]	refers to integral of waveform specified by X1 INPUT FILENAME field
ix2[index]	refers to integral of waveform specified by X2 INPUT FILENAME field
iy[index]	refers to intregral of previous output

Recursive access of previously calculated y output, for example y[n–M], will behave correctly if M <= 4086. If M > 4086, then recursive input is undefined.

Upon beginning a difference equation, all y[n] values are initialized to zero. Thus, for the first access of as yet uncalculated y output, a zero is returned, allowing y[n] to be treated as an "accumulator" if needed. For example, the equation

y[n] = y[n] + n

returns zero for the first output sample calculated (and is equivalent mathematically to a summation of integers, depending on the value of the # OF SAMPLES field; see Difference Equation Menu Fields, below). As a more useful example, the equation

y[m] = y[m] + x1[n]/Fr

would calculate the average value in each frame of the input waveform x1, and create an output waveform containing a sequence of these average values. The total number of values in the output waveform would equal the number of frames of size Fr contained in x1.

Limited Imaginary Operator

A limited imaginary operator "j" can be used in difference equations. The operator is limited because it does not actually represent $\sqrt{-1}$. However, entry in the form:

a+jb

or

a+j*b

where "a" and "b" are standard difference equation expressions, will cause "a" to be written to the real component of the output sample in a complex waveform, and "b" to the imaginary component of the output sample. Use of the j operator always causes the output waveform to be complex.

Nomenclature and Syntax

All nomenclature and syntax used in Hypersignal difference equations is based on traditional DSP textbook notation.

Brackets [] are used to denote array input, where arrays can be external waveforms and internal functions such as unit-step. Brackets can also be nested, with an array access inside another array access.

Parentheses () are used to override the standard level of precedence, if required. Expressions can be nested as many levels deep as required through the use of ().

Number of Output Samples

In all equations a "y[n]=" is assumed for the left side of the equation, unless expicitly entered as something different. When the Difference Equations function executes, the sample index starts at zero and begins incrementing by one. This continues until the *index expression* for the y array output of the equation reaches the value specified in the # OF SAMPLES field (see below). This is very straightforward for a output expression such as "y[n]", but not so simple for an expression such as "y[m]", in which we must wait for the *frame index* to reach the # OF SAMPLES value. In this case the *sample index* must actually increment # OF SAMPLES*FRAMESIZE times. Thus, if you need more output samples than your input waveform(s) is long, you must be sure to specify # OF SAMPLES accordingly.

Operators, Constants, and Functions

All operators, constants, and functions available for use in difference equations are listed below by level of precedence, from highest to lowest:

EXPONENTIATION OPERATOR

^ — The ^ symbol causes operands in an expression such as 2^n to be evaluated as 2^n. Some examples:

```
n^0.5
(-1)^n
x1[n]^(-3)
```

ARITHMETIC OPERATORS

***, /, \, %, +, -** — Multiply, divide, integer divide, modulus, add, and subtract. All * and / operations are evaluated from left to right, then all % operations, then all + and - operations. Integer divide, e.g. "a \ b", is calculated as TRUNC(a/b), where "TRUNC" indicates truncation. Modulus, e.g. "a % b", is calculated as FRAC(a/b)*b, where "FRAC" indicates fractional part.

NON-ARITHMETIC OPERATORS

<,>	The "<" and ">" characters are MIN and MAX operators, respectively. For example, "a<b" will be evaluated as the minimum between of a and b. "a>b" will be evaluated as the maximum between a and b.

FRAME OPERATOR

,	A comma causes operands to be read as "frame,offset". For example, "2,4" means sample 4 in frame 2. The number of samples in a frame is set by the FRAMESIZE field. "offset" is always calculated modulo with respect to FRAMESIZE, forcing offset to always be between 0 and FRAMESIZE-1, inclusive.

CONSTANTS

pi	returns π, where π=3.14159265... when referred to in an equation
Fs	returns the value of the SAMPLING FREQUENCY field when referred to in an equation
Fr	returns the value of the FRAMESIZE field when referred to in an equation
j	'j' or 'j∗' entry causes the remaining part of the difference equation to be treated as the imaginary component of a complex output sample

INPUT ARRAYS

In the list of possible input arrays below, "exp" can be any integer or real expression.

x1[exp]	returns input from the waveform specified in the X1 INPUT FILENAME field
x2[exp]	returns input from the waveform specified in the X2 INPUT FILENAME field
y[exp]	returns recursive input from previously calculated output values
u[exp]	unit-step function: equal to 1 for exp≥0, equal to zero otherwise
d[exp]	delta function: equal to 1 for exp=0, equal to zero otherwise

TRIG, TRANSCENDENTAL, AND CALCULUS FUNCTIONS

ABS(arg)	absolute value of arg
ACOS(arg)	returns, in radians, arcosine of arg
ASIN(arg)	returns, in radians, arcsine of arg
ATAN(arg)	returns, in radians, arctangent of arg
COS(arg)	cosine of arg; arg is expected to be in radians
COT(arg)	cotangent of arg; arg is expected to be in radians
Darray[exp]	returns derivative of array[exp] value, where "array" can be 'x1', 'x2', or 'y'

EXP(arg)	e^{arg}, where e=2.71828182...
FRAC(arg)	returns fractional part of real value
Iarray[exp]	returns integral of array[exp] value, where “array” can be ‘x1’, ‘x2’, or ‘y’
INT(arg)	removes fractional part of real value
LOG(arg)	$\log_{10}$(arg)
LN(arg)	natural log of arg
RAN(arg)	returns pseudo-random real number between 0 and arg, inclusive
RND(arg)	rounds arg to nearest integer
ROUND(arg)	rounds arg to nearest integer
SIGN(arg)	returns -1 for arg, 1 for arg=0
SIN(arg)	sine of arg; arg is expected to be in radians
SQR(arg)	arg^2
SQRT(arg)	$\sqrt{arg}$
TAN(arg)	tangent of arg; arg is expected to be in radians
TENX(arg)	10^{arg}
TRUNC(arg)	truncates arg to next lowest integer

Difference Equations Menu Fields

The following data-entry menu fields may be modified and entered after selection of the Difference Equations function:

FILENAME:

This field is used to specify a filename which is used both for the input equation text file and also for the output waveform file.

Equation text files are saved (filename extension “.EQU”) whenever you execute a difference equation for later recall and modification. Besides the actual equation, the equation text file also includes all other data-entry menu parameters.

An output time domain waveform (“.TIM” extension) is assumed unless an extension is entered explicitly. For example, to create a frequency domain magnitude waveform file as output, you might enter ‘equ_name.FRQ’.

If the input equation file is not found on the current drive and subdirectory shown in the User Setup area, a list of “.EQU” files will be displayed in the System Status area. You can use the ‘PgDn’ key to page through the pop-up directory display, and the ‘PgUp’ key to return to the first page of the directory. The search strategy for the pop-up directory display will change with consecutive use of the ‘PgUp’ key—see section 3.4.2, Pop-Up Directory Display, for details.

DIFFERENCE EQUATION:

This data field allows you to enter up to four lines of equation text. The four lines are concatenated to form the complete equation; thus, lines may begin and end in mid-expression, with any character.

For a complete list of allowable operations, functions, and input waveforms, see above.

If an error is detected in the difference equation field (i.e. incorrect format, mismatched parentheses, etc.) an error message will be displayed that describes the detected error.

X1 INPUT FILENAME:

This field allows you to specify an input waveform filename, and then refer to it in the equation as 'x1[exp]', where "exp" represents some combination of indexes, numbers, operators, constants, and functions. Also, "exp" may itself be a function or nested expression of another waveform, such as x1[y[n−SQR(n)]].

An input waveform filename extension of ".TIM" is assumed unless you explicitly enter a different extension. For example, to use an imaginary component frequency domain waveform file, you would enter 'filename.IMA'.

X2 INPUT FILENAME:

This field allows you to specify a second input waveform filename, and then refer to it in the equation as 'x2[exp]', where "exp" represents some combination of indexes, numbers, operators, constants, and functions.

An input waveform filename extension of ".TIM" is assumed unless you explicitly enter a different extension. For example, to use a power spectra frequency domain waveform file, you would enter 'filename.PWR'.

OF SAMPLES:

This field should specify the number of total samples in the output waveform. An entry of '0' (zero) indicates that all input waveform samples should be used (the maximum between the x1 and x2 waveforms will be used if both are referenced in the equation). A '0' entry is not allowed if x1 and x2 are not referenced in the equation.

NOTE: **If the left side of the equation is not explicity specified, then a "y[n]=" output is assumed. Thus, if you desire the output to be other than y[n] (e.g. y[m]), you must enter it along with an "=" sign. If you do this, and the y output array index expression (e.g. "m") array advances more slowly than the sample index "n",** ***then you must explicitly specify the*** **# OF SAMPLES** ***field in order to accurately control the output waveform length.***

SAMPLING FREQUENCY:

This field should specify the sampling frequency, in Hertz, for the output waveform. This value can be referred to as 'Fs' in the equation.

NOTE: **The SAMPLING FREQUENCY field is not actually used when evaluating a difference equation, other than references to the "Fs" constant inside the equation. However, in cases where an x1 or x2 waveform is referred to in the equation, one or both of the x1 and x2 sampling frequencies should agree with SAMPLING FREQUENCY for subsequent processing and display. A '0' (zero) entry will automatically cause the x1 or x2 sampling frequency to be assigned to the output waveform (x1 has precedence over x2 if both are referenced in the equation). A '0' entry is not allowed if both x1 and x2 are not referenced in the equation.**

FRAMESIZE:

The FRAMESIZE data field is used to specify the framesize of the output waveform for subsequent operations (display, FFT, etc). If '0' is entered, the framesize values of x1 or x2 will be used if referenced in the equation, with x1 taking precedence. This value can be referred to as 'Fr' in the equation.

Note that the total number of samples being evaluated in the output waveform divided by FRAMESIZE determines the number of frames to be be performed, indicated at the bottom of the display during processing.

FRAMESIZE is the number of samples per frame used when an equation references the "frame index" m.

Some General Notes About Difference Equations

The "FRAME xx OUT OF XX" frame counter displayed during Difference Equations processing is intended ***as a progress indicator only***. The total number of frames is pre-calculated based on # OF SAMPLES ÷ FRAMESIZE, and has no relationship to the current "m" frame value index, which may in fact be much different.

Trigonometric overflow values, such as TAN(π/2) or ASIN(1.1), are evaluated as a very large number with a sign the same as that of the nearest valid argument.

Differentiation followed by integration (see below) will not restore a dc bias, if one was originally present—this is equivalent to an arbitrary constant of integration that is not accounted for.

Hypersignal-Macro software can represent waveform file data with magnitude ranging from 2.93873e-39 to 5.575e+42. This is done by storing data values as 16-bit "mantissas" along with an overall exponent value in the waveform header. This storage method allows both extremely fast throughput, especially with real-time DSP/Acquisition hardware, and also fairly good dynamic range. The Difference Equations function produces output waveform data in this format according to the following rules:

1) If the total number of output samples is less than 4032 samples then the data is "adjusted" to maximize the largest mantissa value without exceeding 32767. If any such scaling is performed, then a non-zero exponent value is set in the waveform header.

2) If the total number of output samples is more than 4032 then output is evaluated after the first 4032 samples to see if the maximum |amplitude| was less than 1 or greater than 32767. If so, then scaling is set to assume a worst case (that the data might shrink or expand with continued processing); in this case, scaling/exponent are set to maximum/minimum and minimum/maximum, respectively.

3) If the entire righthand side of the "=" sign (or entire equation if no "=" sign has been entered) is within a 'RND()', 'TRUNC()', or 'INT()' function, specifying integer output, then no scaling is performed, and the output waveform header exponent is set to zero.

NOTE: The above scaling information is included for completeness. Normally you won't be aware of the scaling/exponent relationship inside particular waveform files, since all Hypersignal-Macro functions and displays accept and process waveform data in this format. For more information on the mechanics of the storage process, see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT.

4.6.2 Function Generation with Difference Equations

Below are a few examples of function generation. Please note that there are also several Difference Equations examples in the Tutorial, including step-by-step menu set-ups and resulting waveform displays.

```
y[n] = n*(u[n]–u[n–M]) + y[n–M]
```

The above example generates a sawtooth waveform with a period of M samples. This example can be generalized by replacing the leading "n*" with different F(n)*, where F(n) represents one period of the desired function. For example, an F(n) of

```
A*sin(2*pi*n/M)
```

would create a sine wave generator with amplitude A and period M (in samples). An F(n) of

```
A*(u[n]–u[n–D/100*M])
```

would create a square-wave generator with amplitude A, D% (D percent) duty-cycle, and period M (samples).

Below is an example pseudo-random noise generator, with amplitude=1000 (peak-to-peak), and no dc bias:

```
RAN(1000)–500
```

Note the use of "1000" as an argument to the pseudo-random noise generator: the RAN function always returns a value between 0 and its argument, inclusive. Hence, by subtracting 1/2 arg, pseudo-random numbers are produced that have no dc offset.

4.6.3 Operating On Existing Waveforms With Difference Equations

Below are some examples of operating on existing waveforms using the Difference Equations function. Note that the Tutorial also contains several step-by-step examples, including resulting waveform displays.

Running average for a time domain waveform:

y[n] = y[n–1] + x1[n] – x1[n–Fr]

Adding the square of one waveform to another:

y[n]=x1[n]+SQR(x2[n])

A one-pole lowpass filter:

y[n]=0.75∗x1[n]+0.25∗y[n–1]

Odd-sample decimation:

y[n]=x1[2∗n+1]

DTMF (Dual-Tone Modulated Frequency) Generation:

y[n]=SIN(2∗pi∗n∗A/Fs)+SIN(2∗pi∗n∗B/Fs)

where “A” is the desired frequency for tone 1, “B” is the desired frequency for tone 2, and “Fs” is the sampling frequency.

Noisy sine wave (fundamental frequency=1000 Hz, “Fs”=sampling frequency):

y[n]=5000∗SIN(2∗pi∗n∗1000/Fs)+RAN(200)

Spectral inversion (relative to Nyquist freq):

y[n]=x1[n]∗(–1)^n

N-frame power-spectra running average:

y[m/N),n]=x1[(m%N+m/N),n]+y[(m/N),n]

Summing every FRAMESIZE input points into a new output point:

y[m]=y[m]+x1[n]

Power-spectra averaging for input frequency domain magnitude file (“.FRQ” extension), single output frame:

y[0,n] = x1[m,n] + y[0,n]

Multiplying every frame in a waveform by step function:

y[m,n] = u[0,n–10]∗x1[m,n]

Multiplying a time domain waveform by a complex exponential to produce an in-phase real component and a quadrature imaginary component, where Fc=cutoff frequency, and Fs=sampling frequency:

y[n] = x1[n]∗cos[2∗pi∗Fc∗n/Fs] – j∗x1[n]∗sin[2∗pi∗Fc∗n/Fs]

For a more in-depth look of manipulating frequency domain waveforms, consider the following example:

DIFFERENCE EQUATION:	4096*ATAN(x2[n]/
cont:	(x1[n]+1e-20))
cont:	
cont:	
X1 INPUT FILENAME:	parksm.rea
X2 INPUT FILENAME:	parksm.ima
# OF SAMPLES (0=ALL):	0
SAMPLING FREQ (Hz):	8500
FRAMESIZE:	256

The above example will create a phase response for the "parksm" waveform in a similar way that the FFT Generation function does (except that the FFT Generation function performs a four-quadrant arctangent function, accounting for the signs of the numerator and denominator prior to division, while the Difference Equations arctangent considers only the sign of the division result, and returns only values in the range of $\pm\pi/2$). Note that the ".PHZ", ".REA", and ".IMA" extensions must be entered explicitly in order to override the default ".TIM" extensions. Also note the use of a very small "epsilon" (ε=1e-20) in the denominator of the division operation in case the x1 waveform should happen to contain zero values.

4.7 Using DSP/Acquisition Hardware with Hypersignal

(Real-Time Software Only)

Hypersignal includes six basic functions which can utilize DSP/Acquisition hardware:

1) Continuous waveform acquire and generate to/from disk (Analog Conversion)

2) D/A output from waveform displays

3) Stimulus & Response

4) Digital Oscilloscope

5) Spectrum Analyzer

6) DSP Accelerator

Hypersignal supports many different types of DSP/Acquisition hardware. Supported hardware includes DSP-based boards and "analog-only" boards with no DSP chip. DSP-based hardware usually includes one or more analog channels, either provided onboard or via a digital connection to an external unit. Analog-only hardware generally offers multiple analog channels, higher maximum sampling rates, and more advanced analog signal conditioning features.

For the Analog Conversion, Digital Oscilloscope, and Spectrum Analyzer functions, use of DSP-based hardware allows more options to be active in the software. "Snap-in" digital FIR and IIR filters, FFT analysis, data level-based triggering, and digital gain and offset, can all be performed in real-time. Even with this greater functionality, it still may be required to use analog-only hardware, simply because of the need for high transfer rates or analog signal conditioning that simply cannot be performed in real-time (yet).

For highly detailed information about specific hardware, please consult the Hardware Reference Guide. To select and configure specific hardware in Hypersignal, use the System Configuration menu (see section 2.5.2).

Continuous Waveform Acquisition/Generation

Continuous waveform acquisition and generation to/from disk, D/A output from waveform displays, and Digital Oscilloscope are always active regardless of the whether the hardware is DSP-based or "analog-only" (no DSP chip). The other functions are active only with DSP-based hardware. However, in some cases, adding a small, inexpensive "DSP coprocessor" card will make the Spectrum Analyzer function also operational for analog-only hardware.

D/A Output and Stimulus & Response

The Analog Conversion function allows waveforms to be stored continuously to disk directly from A/D input, and generated continuously from disk directly to D/A output. Parameters such as sampling rate, number of channels, number and type of trigger channels, trigger parameters, "snap-in" real-time filters, gain, attenuation, offset, and amount of data can be specified.

D/A output from waveform displays and stimulus & response are actually subsets of the Analog Conversion function. D/A output from waveform displays basically relies on a graphical interface to specify the output data as opposed to a data-entry menu interface. Stimulus & response is invoked by entering a "stimulus" output waveform filename in the Analog Conversion menu, which causes the stimulus waveform to be sent to the D/A output at the same time an input waveform is being continuously stored to disk. Continuous storage can begin before or after onset of the stimulus. By testing an unknown system with a known stimulus, its response can then be analyzed to learn the system's frequency response, transfer function, delay, etc.

Instrumentation Displays

Instrumentation functions start by requiring some data-entry field set-up, and then present an interactive, continuously updating display as the user-interface. The interactive displays contain controls and options that are, in many cases, derived from traditional, stand-alone oscillscopes and analyzers. They also contain many features that take advantage of the host personal computer environment not found in stand-alone instruments, including: overlaid display of waveform files and analog input, capture of display traces to disk for subsequent DSP processing, screen-dump and publication-quality print, and control over video resolution and screen colors.

DSP Acceleration

DSP acceleration refers to speeding up some time domain functions using DSP-based hardware. By downloading raw data and uploading processed results, large speed increases can be obtained compared with the host 80x86 and 80x87 processors.

Using DSP/Acquisition Hardware

When an operation involving DSP/Acquisition hardware is invoked, two levels of software are at work: the "high-level" host computer software, which you see in the form of user-interface and graphical display, and, for DSP-based boards, the "low-level" DSP software which executes on the DSP chip itself. The high-level software also includes a "device driver" for the particular board, which programs the board with instructions, communicates directly with the DSP software if the board contains a DSP chip, or both. DSP software communication takes place through shared DSP board memory and/or I/O-mapped registers.

When an operation with DSP/Acquisition hardware is first invoked, Hypersignal does the following:

1) Queries the board to see if it is responding correctly at the specified I/O and/or memory base address.

2) If the board is DSP-based, determines the amount of program and/or data DSP memory available for use with data buffers, stimulus storage, etc.

3) Programs I/O-mapped, memory-mapped registers, or DSP memory locations with instructions for DMA, number of channels, gain, sampling rate, FFT size, framesize, buffer base addresses and size, etc.

4) If the board is DSP-based, downloads the appropriate DSP software necessary to allow the DSP to perform the requested operation. Normally, the DSP software is in an executable form created by the link utility provided by the DSP chip manufacturer.

5) Begins execution of the operation—turns the board loose.

6) Continues the operation as specified by the user, transferring data and continuously maintaining communications with registers on the board and/or DSP software.

For DSP-based hardware, modification of the DSP software is encouraged in the Hypersignal environment. Modifications might range from small changes such as adding analog channels, to simulation of entire algorithm or product in real-time. For this purpose, various "DSP Source Code Interfaces" for several different DSP chips are available that allow you to essentially change what Hypersignal does with DSP-based hardware. Each DSP Source Code Interface contains a library of optimized assembly language DSP and math routines, real-time interrrupt structures, and frame-based and sample-based processing structures. Each interface supports several different boards from different manufacturers that use the same DSP chip. Without any changes, linking a DSP Source Code Interface and producing a complete executable file would produce exactly the same executable file that is distributed with Hypersignal software. However, the high-level code acts relatively "dumb"—it only transfers data when told to do so by the DSP, blindly accepts which channels and the total number specified, and unquestioningly writes out uploaded DSP results to disk file or graphical display. Thus, the DSP software can be changed to a great extent to accomplish different things. Some typical examples:

1) add more analog channels, transfer the data to high-level software in interleaved form

2) apply processing (e.g. averaging) to time domain data before storage to disk

3) apply triggering based on some other criteria (e.g. cross-correlation, searching for a "sync" pattern)

4) install an algorithm, use "two-channel" acquisition mode to send up raw data to one display trace, algorithm output data to another

5) replace an accelerated time domain disk-based function with another; for example, replace FFT processing in DSP software with a chirp-Z transform

Please note that there is extensive information on DSP Source Code Interfaces, and modifying DSP source code, in the Hardware Reference Guide.

4.7.1 Making Analog Measurements

Hypersignal offers four basic "instruments" with which to make analog measurements:

1) Digital Oscilloscope

2) Spectrum Analyzer

3) Digital Record/Capture

4) Stimulus & Response

The first two functions are separate menu selections that require data-entry field set-up, and then present an interactive, continuously updating display as the user-interface. The interactive displays contain controls and options that are, in many cases, derived from traditional, stand-alone oscillscopes and analyzers. They also contain many features that take advantage of the host personal computer environment not found in stand-alone instruments, including: overlaid display of waveform files and analog input, capture of display traces to disk for subsequent DSP processing, screen-dump and publication-quality print, and control over video resolution and screen colors.

The third function is really the Analog Conversion menu selection. By using this function to capture arbitrary-length sequences of data in real-time from a system under test, including pre-event and post-event history data, detailed analysis can then be made in non-real-time. Compared to stand-alone instruments that are limited in the amount of data they can store, or force you to trade a lower sampling rate for more data, this approach can offer tremendous advantages.

The fourth function is a subset of the Analog Conversion menu selection. By entering a "stimulus" output waveform filename, a stimulus waveform can be sent to the D/A output at the same time an input waveform is being continuously stored to disk. Continuous storage can begin before or after onset of the stimulus. By testing an unknown system with a known stimulus, its response can then be analyzed to learn the system's frequency response, transfer function, delay, etc.

Digital Oscilloscope

The Digital Oscilloscope function displays incoming analog signals in real-time or near real-time, depending on the current timebase value. Time waveforms from disk files can also be displayed or overlaid in "real-time", allowing mixed-display of live analog input and previously stored waveform data. Parameters under user control include variable timebase (sec/div, msec/div, etc.), amplitude scaling (Volts/div), offset (vertical position) control, and trigger configuration (level, channel) for both single and dual-trace (independently variable) display, and display formats such as line or point mode, time units, and axes.

Spectrum Analyzer

The Spectrum Analyzer function allows dynamic display of frequency domain components, using either analog signal acquisition hardware or previously stored waveform files as input. Intrachannel or interchannel combinations of analog input and file input may be displayed in magnitude, phase, time, power spectra, or transfer function formats. Dual-trace displays can be separate or overlaid. Magnitude and power spectra amplitude formats include linear, log, and negative. Frequency axes can be linear, log (arbitrary base, number of cycles), or octave band (1/N display). For analog input, time domain framesize, FFT order, real-time sampling frequency, and time domain window (rectangular, Hamming, Hanning, Blackman, Bartlett, etc.) may be specified.

Analog Conversion

The Analog Conversion menu selection encompasses both stimulus & response and digital record/capture. This function enables conversion of analog to digital data (A/D) and digital to analog data (D/A) in real-time (continuously, with no loss of data) to and from disk. Acquired waveform files are then available for later analysis, DSP processing, display, and eventual reconversion back to analog form. Since transfer is continuous, waveform file length is limited only by the capacity of the storage media

in use. Analog conversion parameters such as channel selection, real-time filters, signal gain/attenuation, multicondition trigger type and threshold, sampling frequency, time domain analysis framesize, and signal offsets are under user control.

4.7.2 Using Instrumentation Displays

When using instrumentation display, it is important to realize that the DSP always stays in real-time, but the video display may not. The instrumentation displays simply show successive frames uploaded from the DSP board as fast as possible—if they miss, that data is thrown away. Predicting when the video display will and won't be in real-time is highly problematic, because it depends on so many factors, including the amplitude of data (takes longer to draw), number of data points in a frame (also takes longer to draw), speed of video hardware, speed of system I/O bus—well, you get the idea. Some suggestions for increasing the rate of video refresh:

1) Use the "point mode" display option. Point mode is significantly faster than line mode, since no time is wasted connecting samples, but is less readable—especially for "busy", or irregular waveforms.

2) Disable grid display.

3) Make sure your system I/O bus transfer rate setting is at its maximum (some systems have switches for this on the motherboard). First, however, consult the Hardware Reference Guide to ensure that your DSP/Acquisition hardware can tolerate a faster speed.

4) Make sure the 80268/80386 INSTRUCTION SET field in the System Configuration menu is set to 'Y'—again consult the Hardware Reference Guide to ensure that this is legal for your DSP/Acquisition hardware.

5) Make sure your video hardware is set for 16-bit mode transfers.

6) Relocate your system BIOS in RAM, to speed the processing time of video calls.

NOTE: The instrumentation displays will look best if used with video hardware that has dual-page capability. Generally included in this category are EGA-compatible boards, Hercules-compatible boards, and a few non-standard boards whose manufacturer's explicitly claim dual-page capability. When dual-page video hardware is available, Hypersignal uses a "page-switching" scheme, where the display appears to update instantaneously and moving signals appear "smooth", or animated. This is possible because the software can write into one graphics page while displaying another. When only non-dual-page video hardware is available, the software must erase and rewrite the same graphics page that is being displayed, resulting in what appears to be "flicker", i.e. a visible retrace time dependent on the speed of the host machine. If the host machine video hardware is EGA-compatible, but your current video mode resolution (see section 2.4.1, Selecting a Graphics Driver) is higher than EGA (VGA 640x480, or "SuperVGA" 800x600), then Hypersignal will temporarily reduce the video resolution to EGA (640x350) during instrumentation display operation and restore it when finished.

When using dual-trace or overlaid-trace display with analog input and file input (or two files), please note that the waveform file data is scaled as if it came from the analog hardware.

Analog input hardware must be specified by the "System Configuration" menu selection—see section 2.5.2.

The following sections explain how to use the instrumentation displays in Hypersignal.

4.7.2.1 Instrumentation Input

The TRACE1 and TRACE2 fields described below control the input sources for instrumentation displays. Also, real-time filters (and quantization values for fixed-point DSP devices) can be specified in these fields.

Input sources for the Spectrum Analyzer and Digital Oscilloscope functions can be analog input channels carrying data from the currently installed DSP/Analog hardware or standard Hypersignal waveform files. Since the sources are varied, reference is made throughout the Hypersignal documentation to "traces", implying a physical-to-trace mapping, where data is coming from physical sources and being updated continuously in one or two of two possible display traces. Data can be displayed as single-trace (display of trace 1), dual-trace (half-area display of both trace 1 and trace 2), or overlaid-trace (full-area, overlaid display of both trace 1 and trace 2).

Mixed display of analog input and waveform file input is possible; analog display from different channels (referred to as "interchannel") or from the same channel ("intrachannel"). The following combinations of input sources are allowed:

one analog trace
two analog traces
one file trace
one file trace and one analog trace
two file traces

NOTE: When displaying both analog and waveform file input, the waveform file input must be specified for trace 1 display.

Like intrachannel analog input, dual-display from file input can refer to the same file, but this type of operation is not often used. However, intrachannel analog input can be especially useful for viewing the same analog data simultaneously in different formats (e.g. time domain and frequency domain), or when applying real-time digital filters (filtered data vs. unfiltered data).

TRACE1:

This field should specify a waveform file or analog input channel to be used as source 1, and possibly displayed on trace 1.

To specify a waveform file, enter the filename as with other Hypersignal functions. In the Digital Oscilloscope, a ".TIM" filename extension is assumed. In the Spectrum Analyzer, the filename extension is based on the type of input specified in the FORMAT1 and FORMAT2 fields, below. If a waveform file cannot be found, a directory of files will appear in the System Status area.

To specify analog input, enter 'An', where n is the desired input channel on the hardware currently selected. Values for n range from zero to 63.

NOTE: *For analog input only,* a "snap-in" real-time FIR or IIR filter can be specified by entering the filter coefficient filename after the 'An' input channel entry (use a comma as a separator). A quantization value can also be specified (use another comma). See section 4.9.3, Using "Snap-In" Real-Time Filtering.

There are many limitations concerning real-time filtering, which usually depend on the DSP chip being used, the sampling rate, and the length, or size, of the filter. For further explanation of these potential problems, and section 4.9.3, Using "Snap-In" Real-Time Filtering.

Also, there are several step-by-step examples of using real-time filtering in the Tutorial. See Tutorial chapter 11, SPECTRUM ANALYZER.

Normally, all Hypersignal DSP Source Code Interfaces (and the executable versions provided with standard distribution software) are configured for a maximum of 2-channel input. However, you can modify the DSP Source Code Interface for the appropriate DSP chip directly to add additional or different analog channels. Such modifications are relatively easy to make; see section 4.7, Using DSP/Acquisition Hardware with Hypersignal for more information.

TRACE2:

This field should specify a waveform file or analog input channel to be used as source 2, and possibly be displayed on trace 2. If left blank (use 'Delete' or 'Backspace' keys, or 'Enter' key with cursor at start of field), only TRACE1 will be active.

Entry is the same as for the TRACE1 field.

When dual-displaying a waveform file and analog input channel, *the* TRACE2 *field must contain the analog input entry.*

NOTE: Specifying the same analog channel in the TRACE1 and TRACE2 fields is generally treated by the DSP source code as two different channels which happen to contain the same data, requiring equal memory and processing resources. This allows filtered vs. unfiltered display, and different display formats of the same data. However, double the amount of DSP memory is required as for the same operation with one analog channel; see the hardware-specific "Onboard Memory Usage" sections in the Hardware Reference Guide for more information about available DSP memory.

4.7.2.2 Spectrum Analyzer Data Formats

The FORMAT1 and FORMAT2 fields described below control the type of displayed data format for the Spectrum Analyzer function. These fields are omitted from the Digital Oscilloscope data-entry menu because a time data format is always assumed.

FORMAT1 (F1):

This field should specify the type of data display for trace 1 display of the waveform file or analog channel specified in the TRACE1 field above. Possible entries are:

'T' Enables time display for *analog input only*, which may used to synchronize Fourier analysis to periodic characteristics in the data using the trigger level controls during display. This type of display is a time domain format.

'M' Enables magnitude display for either analog input or previously stored file waveforms. Entry may optionally be followed by a number from 0 to 99 ('nn' entry) that specifies the number of frames to average. '0' or '00' entry indicates no averaging. Entry may also be followed by '+' or '-' log display suffixes (the '-' log, or "negative log", suffix inverts the log display). This type of display is a frequency domain format.

Example entries could be: 'M', 'M+', 'M50+', and 'M10'. *If 'nn' entry is specified and the corresponding TRACE field contains a waveform filename, then the software will search for a power spectra waveform file with filename extension ".PWR", which must already exist (i.e. Power Spectra Generation function must have already been executed on the waveform).*

NOTE: For boards with a 16-bit fixed-point DSP device you will only be able to see log magnitude precision to about -67 dB. This is because Hypersignal currently does only a single-precision FFT on these devices.

'H' Enables transfer function display, which is calculated as the FFT output of the input data specified in the TRACE1 field divided by the FFT output of that specified in the TRACE2 field. Entry may optionally be followed by a number from 0 to 99 ('nn' entry) that specifies the number of frames to average. '0' or '00' entry indicates no averaging. Entry may also be followed by '+' or '-' log display suffixes (the '-' log, or "negative log", suffix inverts the log display). This type of display is a frequency domain format.

In the case of analog input specified in both the TRACE1 and TRACE2 fields, the complex ratio of TRACE1 FFT output over TRACE2 FFT output is calculated. If the TRACE1 field specifies analog input and the TRACE2 field waveform file input, then the software will assume that the "opposite channel" in a typical two-channel system should be used for the denominator complex FFT output. Regardless of the analog channels specified or assumed in the TRACE1 and TRACE2 fields, the calculation always represents TRACE1/TRACE2. If the corresponding TRACE field specifies waveform file input then the magnitude waveform file (or power spectra waveform file if 'nn' entry is present) will be displayed *as if the waveform data resulted from a transfer function calculation*. This is an important distinction: no actual transfer calculation is taking place involving the waveform file data, only transfer function display scaling, axis labeling, etc.

Note that if a log magnitude format is enabled ('+' or '-' suffix after 'H' or 'Hnn' entry) onscreen scaling for transfer function display is different than standard log maginitude display. For transfer function display, the default zero dB point is shown as the middle of the trace display, rather than the top of the trace display. Also, subsequent use of the PgUp or PgDn digital scaling functions expands and contracts the vertical dynamic range about the zero dB axis. Linear magnitude format for transfer function display is not usually recommended, as systems with gain <= 1 will produce results to small to see.

Note also that a transfer function trace can be displayed by itself even if both TRACEn fields are

non-blank. This can be accomplished in several ways: by using 'N' entry in the Fn field you want to disable, by entirely blanking the Fn field you want to disable, or by specifying only the trace you wish to see using either '1' or '2' entry in the FORMAT subfield of the TOPOLOGY (TPL) field; see section 4.7,2.6, Instrumentation Display Options, below, for detailed instructions.

'I' — Enables impulse response display, which is a time domain format, and which results from an inverse FFT of transfer function data. Entry may optionally be followed by a number from 0 to 99 ('nn' entry) that specifies the number of frames to average. '0' or '00' entry indicates no averaging. This type of display is a time domain format.

Impulse response display, by definition, requires transfer function calculation. In the case of analog input specified in both the TRACE1 and TRACE2 fields, the complex ratio of TRACE1 FFT output over TRACE2 FFT output is calculated. If the TRACE1 field specifies analog input and the TRACE2 field waveform file input, then the software will assume that the "opposite channel" in a typical two-channel system should be used for the denominator complex FFT output. Regardless of the analog channels specified or assumed in the TRACE1 and TRACE2 fields, the calculation always represents TRACE1/TRACE2. If the corresponding TRACE field specifies waveform file input then an error message will be displayed, since the Spectrum Analyzer does not accept time domain waveform file input.

Note also that a impulse response trace can be displayed by itself even if both TRACEn fields are non-blank. This can be accomplished in several ways: by using 'N' entry in the Fn field you want to disable, by entirely blanking the Fn field you want to disable, or by specifying only the trace you wish to see using either '1' or '2' entry in the FORMAT subfield of the TOPOLOGY (TPL) field; see section 4.7,2.6, Instrumentation Display Options, below, for detailed instructions.

'PH' — Enables phase display for *analog input only*. May optionally be followed by a number from 0 to 99 ('nn' entry) that specifies the number of frames to average. '0' or '00' entry indicates no averaging. This type of display is a frequency domain format.

'Pnn' — Functions in same manner as 'Mnn' entry above, but is allowed to maintain consistency with other, file-based frequency domain displays that use a 'P' entry to specify power spectra display. This type of display is a frequency domain format.

'N' — None. Causes the corresponding trace display to be disabled. Other entry ('nn' averaging, '+' or '-' log suffixes) after the 'N' is ignored.

NOTE: To see simultaneous display of data in different formats, or to compare filtered and unfiltered data, *from the same analog channel,* 'An' must be entered in *both* the TRACE1 and TRACE2 fields (i.e. 'n' must be the same).

FORMAT2 (F2):

Entry is identical to the FORMAT1, above. If the TRACE2 field has been left blank, specifying no input on trace 2, then entry in this field is ignored.

4.7.2.3 Digital Oscilloscope Triggering

The TRIGGER TRACE,LEVEL field below is not included in the Spectrum Analyzer function, because of lack of data-entry menu space. However, the standard interactive trigger commands, using the '+' and '-' keys, are active in both instrumentation displays. See section 4.3.1, Interactive Waveform Display Commands, for a description of these commands.

TRIG TRACE,LEVEL:

Specifying a '1' or '2' in the first portion of this field selects the initial display trace input to act as "trigger trace". Once inside the display, the interactive TRIGGER TRACE command ('T') key can be used to toggle the trigger trace between trace 1 and trace 2.

After the trigger trace, an initial trigger level can also be specified. By entering a level (in Volts) after the initial trace value (e.g. '1,1.5'), the initial trigger level upon entry to the display function will be set to this value. This allows very precise control of single-shot triggering and other exact triggering situations.

If no LEVEL entry is made, default trigger levels are zero, which effectively disable triggering (equivalent to the "AUTO" mode on a traditional analog scope).

Note that the initial trigger level cannot exceed the value of the INPUT RANGE field (section 4.7.2.5, below).

NOTE: Some analog-only hardware have no onboard DSP processing capability *and* no programmble trigger-level hardware circuitry (or may have only a "gate", such as a TTL-level external input). In these cases, the trigger function is disabled inside instrumentation displays. In a few cases, the trigger function is not completely disabled: incrementing the trigger level a few times (see TRIGGER LEVEL interactive commands in section 4.3.1) may enable the external TTL-level trigger, if the board is so equipped. See the Hardware Reference Guide for information on this subject for specific hardware.

4.7.2.4 Instrumentation Analog Sampling Frequency

The ANALOG SAMPLING RATE field described below controls the analog hardware sampling frequency only, and has no effect on waveform *file* display, regardless of the sampling frequency used when the waveform was originally acquired.

ANALOG SAMPLING RATE:

This field should specify the analog hardware sampling frequency to be used. Consult both the hardware manufacturer's documentation and the Hardware Reference Guide for limits on minimum and maximum sampling frequencies for the DSP/Acquisition hardware in use. Also, see section 4.9.3, Using "Snap-In" Real-Time Filtering, for a discussion of the effect of real-time filters on maximum sampling frequency.

This field is ignored if the TRACE1 field specifies a previously stored waveform file.

NOTE: Always enter the actual sampling frequency desired, regardless of 1 or 2 analog channel display—any modification of the sampling frequency required prior to programming the hardware or DSP chip will be handled internally by the software, if needed at all.

4.7.2.5 Instrumentation Input Range and Gain

For DSP/Acquisition hardware that does not have programmable hardware gain, the INPUT RANGE field described below should be set to the default input level allowed by the hardware. If these levels can be changed by trimpot or jumpers, you must change the INPUT RANGE field also. For DSP/Acquisition hardware with programmable hardware gain, the INPUT RANGE field allows the current gain level to be set.

INPUT RANGE:

This field should specify the full-scale input range for the hardware currently in use. For example, if the board in use allows a ±10 Volt analog input, then the value of INPUT RANGE should be entered as '10'. For hardware that has programmable-gain stages or programmable full-scale ranges, this field can be used to control the current gain setting (see the Hardware Reference Guide to see if this is the case and what range is allowed).

NOTE: For DSP-based hardware, regardless of how this field is used, digital "gain"—multiplication scaling implemented in DSP software—is also available during display, using the interactive VERTICAL SCALING commands ('PgUp' and 'PgDn' keys).

When controlling the gain for programmable DSP/Acquisition hardware, correct amplitude units display is possible because the software knows the hardware's maximum input A/D values (e.g. 32767 for a 16-bit channel). For example, a board might be programmed so that a gain of 1 allows a full-scale voltage range of ±10 Volts. If you then enter an INPUT RANGE value of '5', the board would be programmed with a gain of two.

One category of programmable hardware gain that warrants further discussion includes boards that use TLC3204x "AIC" (analog interface chip) devices as analog converters. Due to its built-in A/D and D/A, anti-aliasing filters, programmable sampling rate, programmble input source, programmable gain, and simple serial I/O connection to most DSP chips, all contained in a single-chip package, the TLC3204x has become quite popular for use with DSP boards. For these boards, the software downloads

an "AIC control word" that specifies a default gain (normally ±6 Volts) to DSP board memory before run-time. Thus, by changing the INPUT RANGE field, you can select one of the two or three gain settings allowed by the TLC3204x (depending on its input source). For example, changing INPUT RANGE to '3' would double the TLC3204x gain. Of course, besides gain, many other programmable parameters can be changed in the control word. If you need to change aspects of TLC3204x operation that can't be controlled from the data-entry menus, you can alter this value directly in the "HSMACRO.CNF" configuration file. However, if you change the two gain bits in the control word, the software will disable data-entry menu control over gain, assuming that you want to set it directly.

For hardware *without* programmable-gain, you would change INPUT RANGE from the board's default full-scale value *only to match a manual gain adjustment (trimpot or jumper).*

4.7.2.6 Instrumentation Display Options

Display options in the Digital Oscilloscope and Spectrum Analyzer functions are very similar. Entry is controlled through the FORMAT,PLOT,GRID field in the Digital Oscilloscope, and through the latter part of the TOPOLOGY (TPL) field in the Spectrum Analyzer.

FORMAT,PLOT,GRID or TOPOLOGY (TPL):

This field specifies whether whether dual-trace display is actually dual, or overlaid, whether waveform sample drawing is "line mode" or "point mode", and the type of grid display. Entry should be in the form of:

FORMAT,PLOT,GRID

for the Digital Oscilloscope, and

nD,FORMAT,PLOT,GRID

for the Spectrum Analyzer (see section 4.7.2.9 below, Spectrum Analyzer 3-D map mode, for information on the "nD" subfield).

Possible entries for FORMAT are:

'D'	dual-trace display
'S'	subtracted-trace display (assumes two-trace operation)
'O'	overlaid-trace display (assumes two-trace operation); see the Tutorial for examples of overlaid-trace display, including comparison with previously stored waveform file data
'1'	display only trace 1
'2'	display only trace 2

Possible entries for PLOT are:

'LIN'	specifies "line mode", or smoothly connected waveform sample points; line mode slows the maximum rate of screen update somewhat, but usually produces a more readable display
'PNT'	specifies that sample points are plotted but not connected ("point mode")

Possible GRID entries are:

'S'	enables a solid grid
'I'	enables an "intersected" grid; only dots where grid lines would normally intersect are displayed
'N'	turns grid display off (blank entry disables grid display also)

4.7.2.7 Spectrum Analyzer Frequency Axis

LOG FREQUENCY:

This field controls the frequency axis (x-axis) for the Spectrum Analyzer display. This field has no effect on trace displays containing time domain data. A simple 'N' (none) entry in this field, specifies a linear frequency axis, starting at zero (dc) and ending at half the sampling rate (the maximum bandwidth, depending on the anti-aliasing filters used by the DSP/Analog board).

Other entries can be used to specify a base N log axis, or in some cases, special entry such as 1/N octave band. Entry should be in the form of:

base^C,min

where “base” is non-zero to set the number base of the logarithm that will be used for the frequency axis, “C” the number of cycles that will be visible if the frequency axis is log, and “min” the left cutoff value if the frequency axis is log (in Hz).

The Hypersignal default entry is ‘N’ (none), which specifies a linear frequency axis. Although section 4.5.3.1, Log vs. Linear Frequency Domain Display, contains precise and detailed examples of entry for this field, a few examples of possible entry are shown below. For example, the entry below would specify 3 cycles of log 10 frequency axis display, with a minimum frequency axis value of 100 Hz:

LOG FREQ (N/base,Min): 10^3,100

As another example, the entry below would specify 12 cycles of log 2 frequency axis display, with a minimum frequency axis value of 20 Hz:

LOG FREQ (N/base,Min): 2^12,20

To toggle easily between linear and log frequency axis modes, simply make the first letter of the LOG FREQ field entry an ‘N’ or part of a legitimate entry, leaving the rest of the entry in place.

Grid marks in log frequency axis trace displays are shown exactly on multiples of the logarithm base. For example, for a base 10 log frequency axis, the grid marks would appear at: ..., 90, 100, 200, ..., 900, 1000, 2000, ..., 9000, 10000, 20000, ..., etc. However, the maximum frequency axis value (half the sampling rate) and the minimum frequency axis value (default or specified in the Min subfield of the LOG FREQ field) may be any number.

1/N octave band display is created by dividing each octave of a log2 frequency axis into N equal sections. For example, the entry below would specify 1/6 octave band display, with a 20 Hz minimum frequency axis value:

LOG FREQ (N/base,Min): OCT^6,20

A non-fractional octave band display could be entered in the LOG FREQ field as either:

LOG FREQ (N/base,Min): OCT^1,min

or alternatively,

LOG FREQ (N/base,Min): 2^11,min

where “min” affects only the minimum value of the frequency axis, and might typically range anywhere from 2 Hz to 200 Hz.

NOTE: Essentially, octave band boundaries are calculated in the Hypersignal Spectrum Analyzer display by repeatedly reducing the *linear* frequency axis by two, starting with half the sampling frequency (bandwidth). Fractional bands are then calculated by segmenting the octave bands evenly. In linear magnitude display modes, the software calculates an arithmetic mean for each fractional band. In log magnitude modes, the software calculates a geometric mean for each fractional band—this calculation is currently being replaced by an arithmetic mean.

Since a varying number of available FFT output points available for 1/N averaging in each band, the Spectrum Analyzer only draws the resulting averaged value wherever an FFT point is available, so as to depict the frequency resolution of the display clearly. This is why you see “spaces” between points as frequency decreases in Figure 11-13. Note that increasing the FFT length will increase the frequency resolution (and that this can be done independent of the time framesize value). Note also that the crosshair cursor will only land on available FFT points in log frequency axis displays.

4.7.2.8 Spectrum Analyzer Fourier Parameters

The FFT ORDER,FRAMESIZE,3D FRAMES and WINDOW fields control the Fourier analysis parameters that are downloaded to the DSP board for real-time FFT processing. Below is a description of these two fields:

FFT ORDER,FRAMESIZE,3D FRAMES:

This three-part field consists of the FFT length, input framesize (applied prior to FFT), and number of "3D" frames when the "3-D map mode", or "running waterfall" display is active (see section 4.7.2.9 below, Spectrum Analyzer 3-D Map Mode).

FFT length is actually entered as FFT ORDER, where FFT length = $2^{\text{FFT ORDER}}$. Minimum entry is 2, maximum is 12 (although operation for some DSP boards, depending on the amount of DSP memory found by Hypersignal, may dictate a lower maximum entry).

FRAMESIZE can be used to control zero-filling when this value is less than FFT length. Minimum entry is 2 and maximum entry is the current FFT length. See section 4.4.1, Framesize and Overlap Discussion, for more information on choosing appropriate values of FRAMESIZE and FFT ORDER. See section 4.4.2, Zero-Filling and Truncation Discussion, for more information about the effects of zero=-filling.

3D FRAMES is actually not a Fourier analysis parameter. It is not downloaded to DSP boards, and affects both analog input and file input. See section 4.7.2.9 below, Spectrum Analyzer 3-D Map Mode, for information on what this field does and how to enter it.

WINDOW:

This field should specify the type of window applied to each input frame in the time domain prior to FFT processing. Possible entries are:

'R'	rectangular (i.e. none)
'Hm'	Hamming
'Hn'	Hanning
'Bl'	Blackman
'Bt'	Bartlett

Windows are always calculated as being FRAMESIZE (above) samples long and applied to each frame of the input data symmetrically. Thus, if FRAMESIZE is less than the FFT length, the zero samples beyond FRAMESIZE are not included in the windowing process. Also note that when FRAMESIZE is even, the actual "center point" of the window lies on a "half-sample", causing the points FRAMESIZE/2 and FRAMESIZE/2+1 of the window to be equal.

NOTE: The FRAMESIZE, FFT ORDER, and WINDOW subfields apply only to analog input, or when either the TRACE1 or TRACE2 fields (section 4.7.2.1) specify analog input. If all input is from waveform files, these subfields are completely ignored.

4.7.2.9 Spectrum Analyzer 3-D Map Mode

The Hypersignal Spectrum Analyzer function has an additional "3-D map mode", or "running waterfall" display mode. In this mode, successive FFT output frames are displayed slightly behind each previous frame to create the illusion of a "waterfall". The advantage to such a display is that you can see a time history of the frequency characteristics in the analog or waveform file input data. For a discussion on how the choice of FRAMESIZE affects time-varying frequency characteristics, see section 4.4.1, Framesize and Overlap Discussion.

To invoke the 3-D map mode, a '3D' entry must be made in the nD subfield of the TOPOLOGY (TPL) field, and an appropriate number of 3-D frames must be specified in the 3D FRAMES subfield. Possible entries for these subfields are:

nD:

The nD subfield of the TOPOLOGY field in the Spectrum Analyzer data-entry menu controls the display mode. Entries can be:

'2D'	specifies the standard, 2-dimensional, continuously updating instrumentation display
'3D'	specifies the "running waterfall" mode of display, where some number of successive frames (see the 3D FRAMES subfield, below) are drawn, each slightly behind the previous, before the screen is refreshed

3D FRAMES:

This subfield controls the number of frames drawn in the 3-D map mode before the screen is refreshed. Minimum entry is 1, default entry is 40. Large amounts of frames may cause data to be missed, because the video display cannot keep up with the real-time processing rate of the DSP.

Note that the 3-D map mode is intended mainly for frequency data display, and will yield a readable time trace display only if the signal is periodic and has been stabilized via the trigger commands (see section 4.7.2.3, Triggering, above). Although the 3-D map mode will display any format of data and combination of input soruces, best results are usually obtained using single- or dual-trace linear magnitude format, because the noise floor is kept low and unobtrusive in the 3-D display. Use the 'PgUp' and 'PgDn' keys to best size frequency data for 3-D map mode display. Note also that overlaid-trace display, if requested in the FORMAT subfield (see section 4.7.2.6, Display Options, above) will be deactivated in the 3-D map mode, because overlaid-trace display cannot be enabled simultaneously with the hidden-line removal required in the 3-D map mode.

For example, a Spectrum Analyzer data-entry menu such as:

```
          TRACE1 (An=anal):  a0
          TRACE2 (An=anal):  a1
          F1(Pnn/M+/H/T/PH):  m
          F2(Pnn/M+/H/T/PH):  m
          LOG FREQ (N/base,NN):  n
          ANALOG SAMPLING RATE:  44000
          INPUT RANGE (Volts):  5
          FFT ORDR,FRMSIZ,3D FRMS:  10,256,75
          WINDOW (R/Hm/Hn/Bl/Bt):  Hm
          TPL (nD,D/O,LIN/PNT,S/I):  3D,D,lin,i
```

would specify dual-trace display from analog channels 0 and 1, linear magnitude format on both traces, linear frequency axis, a sampling rate of 44 kHz, scaling on the vertical axis to match a ±5V input range on the board, an FFT size of 1024 (order=10, as in 2^{10}=1024), an FFT input framesize of 256 (implying 768 zero-filled points), 75 frames to display in the "time direction" (going into the screen), a Hamming window to be applied to the 256 points prior to the FFT, 3-D topology (which will use the 75 frames; '2D' would ignore and revert to single-frame), "line" mode for plotting, and an "intersected" grid.

Note that you can "pause" the 3-D map mode display at any time using the 'Pause' key on the keyboard. Also, the 'F2' key (see section 4.3.2, Printing Waveform Displays) will print the current 3-D map mode display (but you must be using a "printable" color combination—see section 4.3.2, Printing Waveform Displays, for details).

4.7.2.10 Spectrum Analyzer Amplitude Display and Calibration

Spectrum Analyzer linear magnitude display is always in Volts, with the INPUT RANGE field (see section 4.7.2.5 above, Input Range and Gain) value at the top of the scale.

Log and -log magnitude displays are always in dB, with lower amplitude values displaying as decreasing dB values. For the default value of system calibration offset, "0 dB" is placed at the top of the vertical scale and is defined as the *maximum possible A/D input value*. However, the ENGINEERING UNITS command (see below) can be invoked to assign a different dB value to any arbitrary amplitude value during display.

During both linear and log amplitude display, Hypersignal accounts for the number of A/D converter bits, as well as FRAMESIZE/2/W scaling caused by the FFT, where FRAMESIZE is the time domain input framesize (see section 4.7.2.8 above), and W is a scale factor dependent on the time domain window used. For more information about this scaling, including both the Spectrum Analyzer and other Hypersignal frequency domain displays, see section 4.5.3.2, Frequency Domain Display Scaling.

Calibration of the Spectrum Analyzer for log displays is performed by using the arrow keys or mouse to position the crosshair cursor at the desired level (usually on a peak value being displayed from some "reference" input), and then invoking the ENGINEERING UNITS command ('E' key). You will then be prompted to enter a dB value to assign to the crosshair cursor level.

When you do this, two events occur:

1) The Spectrum Analyzer vertical dB scale immediately changes to reflect the new calibration.

2) A new "system calibration offset" is set.

The system calibration offset can be used in other frequency domain displays to calibrate their display scales exactly as with the Spectrum Analyzer. This is done using the 'C' entry in the REF subfield of frequency domain file-based displays.

Also, perhaps most importantly, *the system calibration offset is stored in the file header of waveforms acquired from DSP/Acquisition hardware in the Analog Conversion function.* This means that acquired waveforms with one calibration offset can be compared with waveforms acquired at a different time with a different calibration offset. This mode of comparison is enabled by a 'W' entry in the REF subfield in frequency domain displays.

For more information about the REF subfield in frequency domain file-based displays, see section 4.5.3.1, Linear vs. Log Frequency Domain Display.

NOTE: The system calibration offset retains its value during program operation until changed. To return to the default calibration, place the crosshair cursor at the very top of a Spectrum Analyzer log display, invoke the ENGINEERING UNITS command, and enter '0'. The system calibration offset is stored in the Hypersignal user-environment configuration save file.

4.7.2.11 Summary of Interactive Instrumentation Display Commands

This section briefly lists all instrumentation display interactive keyboard and mouse commands. For a more detailed explanation of individual commands, see section 4.3.1, Interactive Waveform Display Commands.

BACKGROUND COLOR: 'B' key

The background color command changes the color (or intensity, blinking, or other attribute for a monochrome display) of the background used in the display. This command only affects the screen and not any images sent to a printer (the pop-up printer menu invoked by the 'F2' key has its own background control).

COLOR:

See WAVEFORM COLOR, GRID COLOR, LABEL COLOR, BACKGROUND COLOR commands.

CROSSHAIR CURSOR: arrow keys or mouse motion

Displays crosshair cursor for the *current active trace*. Cursor will not appear if time data is being displayed.

This command is active only in the Spectrum Analyzer display.

ESCAPE: 'Esc' key or leftmost two mouse buttons on 3-button mouse

Pressing the 'Esc' (escape) key at any time will return to the previous data-entry menu, and the data-entry editor will be re-activated.

ENGINEERING UNITS: 'E' key

Pressing the 'E' key while the crosshair cursor is displayed will cause a prompt to appear asking for a dB value to assign to the current cursor vertical position. This value will set the system calibration offset. Please see section 4.7.2.10, Spectrum Analyzer Amplitude Display and Calibration.

This command is active only in the Spectrum Analyzer display.

GRID COLOR: 'G' key

The grid color command changes the color (or intensity, blinking, or other attribute for a monochrome display) of the grid and coordinate axes (if any) in the display.

HELP: '?' or 'F1' keys, rightmost two mouse buttons on 3-button mouse

Displays on-line Help pertaining to the current waveform display function. Use the 'Esc' key again to return to the current graphical display.

LABEL COLOR: **'L' key**

The label color control changes the color (or intensity, blinking, or other attribute for a monochrome display) of all text labels in the display.

OFFSET:

See VERTICAL OFFSET command.

PRINT: **'F2' key**

Invokes the pop-up print menu, which allows both screen dumps and publication-quality prints of the current waveform display to nearly 200 different types of printers. The output device (LPT1, COM1, disk filename, etc.), dimensions, dots-per-inch, orientation, background, etc. can be specified in the pop-up print menu. Publication-quality prints require adequate EMS memory. Please see section 4.3.2, Printing Waveform Displays, for more detailed information.

REDRAW: **'Enter' key**

Redraws the current display using initial settings. All timebase, triggering, and vertical scaling, and vertical offset settings are reset.

SCALING:

See VERTICAL SCALING command.

TIMEBASE: **left and right arrow keys, horizontal mouse motion**

For the Digital Oscilloscope display, the left and right arrow keys (or mouse) can be used to control the current timebase setting, with the right arrow expanding the waveform (showing less time per division), and the left arrow contracting the waveform (increasing the time per division). Timebase control affects both traces in a dual-trace display; i.e. timebase control is independent of the current active trace. Time axis (x-axis) units are always displayed in 2, 5, or 10 units per division, where units can be sec, msec, μsec, etc. depending on the sampling rate.

This command is active only in the Digital Oscilloscope display.

TRACE 1 or TRACE 2: **'1' or '2' keys**

If you are in a dual-trace display, use the '1' or '2' keys to select the "current active trace" for which other interactive commands will apply, such as zoom, trigger level control, crosshair cursor, vertical scaling, vertical offset, etc.

The current active trace is indicated on the display by two small rectangles at the righthand side of the display area. A rectangle filled in ("solid") with the color matching that trace's waveform color indicates the current active trace.

TRACE 1 and TRACE 2 "Lockstep": **'Shift-1' or 'Shift-2' keys**

If you are in a dual-trace display, use the 'Shift-1' or 'Shift-2' keys to cause the vertical offset (time domain data only) and vertical scaling commands to apply to both traces simultaneously. Pressing the '1' or '2' to re-select a specific trace always de-activates Lockstep mode.

TRIGGER LEVEL: **'+' and '-' keys**

Use the '+' and '-' keys to increase and decrease, respectively, the trigger level for the *current active trace.* Triggering used in the Hypersignal instrumentation displays is essentially equivalent to the "NORM" mode on a traditional analog scope, except that the last frame is frozen on the display when trigger conditions are no longer met. Note that trigger level is shown numerically on the display at the lower right as well as in the form of a "pointer" that moves up and down along the vertical axis next to the current active trace. A trigger level value of near zero effectively turns triggering off (equivalent to "FREERUN" mode on traditional analog scope).

Use positive trigger levels ('+' key) to trigger on a signal's rising edge or slope, and negative trigger levels ('-' key) to trigger on a signal's falling edge or slope. "One-shot" type display is easily achieved by pre-adjusting trigger level, then allowing input to

occur. This can be accomplished by selecting the trigger trace ('T' key; see below) to the other trace, adjusting trigger level for the desired trace (assuming it is selected as the current active trace), and then returning the trigger trace to the desired trace when ready. This works because the trigger trace and current active trace selections are made independently. Note that in the Digital Oscilloscope an additional method is available, since initial trigger level can be specified exactly in Volts on the data-entry menu.

NOTE: **In the Digital Oscilloscope function, triggering is operational for both analog input data and waveform file input data. This is because triggering is essentially a time domain operation, where amplitude levels and slopes are being measured. However, in the Spectrum Analyzer function, triggering does not apply to waveform file input, because the Spectrum Analyzer function does not accept time domain waveform files as input, only frequency domain waveform files.**

TRIGGER TRACE: **'T' key**

Use the 'T' key to toggle the current trigger trace between the displayed traces. To see—and vary (using '+' and '-' keys)—the trigger level for a trace, be sure that the active trace is the same as the desired trace. Otherwise, you can change the trigger level, and wonder why it is not affecting your display. The current trigger trace is shown at the lower right of the display.

NOTE: **The current trigger trace is completely independent of the current active trace.**

UNITS: **'U' key**

This command rotates the current unit display (either time or frequency) up to the next standard SI value (e.g. mS to S), then down (e.g. kHz to Hz) then back to the automatic default. If unit overflow or underflow occurs, an error message is displayed and units remain unchanged. Rotation is modulo; pressing the 'U' key three times displays all SI unit possibilities.

VERTICAL OFFSET: **up and down arrow keys, vertical mouse motion**

Use this command to control the vertical position, or offset, of the waveform displayed in the *current active trace.*

In the Spectrum Analyzer display, this command is only active when time domain data is being displayed in the current active trace.

VERTICAL SCALING: **'PgUp' and 'PgDn' keys**

The 'PgUp' and 'PgDn' keys can be used to control vertical amplitude scaling for the *current active trace.* They are active independent of other waveform display settings such as timebase, trigger level, log/linear amplitude, vertical offset, etc.

To scale a waveform up vertically (make it larger), use the 'PgUp' key. To scale a waveform down vertically (make it smaller), use the 'PgDn' key. Note that vertical amplitude units change along with waveform size. Also note that log mode displays (with the exception of transfer function display in the Spectrum Analyzer) "pin" the top of the display and expand/contract the data from the top down. Linear mode displays (includes time domain, phase, and impulse response) expand/contract the data about the x-axis (typically equivalent zero).

WAVEFORM COLOR: **'C' key**

The waveform color command changes the color (or intensity, blinking, or other attribute for a monochrome display) of the waveform in the display.

WRITE TO DISK: **'W' key**

Writes the *current active trace* to a waveform file.

After initially striking the 'W' key, you will be prompted for an output filename. If the output file already exists, you will be prompted whether to overwrite the existing file. See chapter 11 in the Tutorial, SPECTRUM ANALYZER, for several excellent examples of writing trace displays to waveform files in various formats.

NOTE: **This is a Very Important Spectrum Analyzer Note. When using the 'W' command in the Spectrum Analyzer function, keep in mind that any linear/log magnitude or transfer function traces with averaging active *are written out to power spectra files*. Such files have a ".PWR" filename extension, requiring the POWER SPEC field to be set to 'P' in other frequency domain displays to view them. Keep in mind also that averaging, or the 'P' entry, must be specified in the Spectrum Analyzer to retrieve these files for comparison viewing.**

Note also that phase traces are written out to ".PHZ" files, and time data and impulse response data traces are written out to ".TIM" files.

INVERTED FILTER OUTPUT (HYPERSIGNAL-ACOUSTIC ONLY)

In Hypersignal-Acoustic, the WRITE function ('W') key will also prompt you for "inverted filter output". If answered affirmatively, the program will, in addition to the standard output as described above, also create an FIR desired response curve that represents the inverse of the current Spectrum Analyzer magnitude trace. This option is intended to create a real-time filter that can be applied to the input of a system and cause the system frequency response output to be more or less flat.

Below are the basic operational procedure and some items of significance for this function:

1) The inverse of the displayed trace is created by choosing a "zero dB point" (current vertical position of crosshair cursor) for the inverted trace, and then rotating data about this point. You can think of the rotation as a rectangle spinning symmetrically about the zero dB point, where the overall height of the rectangle is the current dB vertical scale of the display. When you display the inverted trace in the filter construction graphical display, the chosen zero dB point will be displayed at the top of the graph. Currently, in the FIR Filter Construction graphical display, if the inverted output has gain greater than one (above 0 dB), then you can only see the entire response with a linear vertical scale (use the 'L' key). However, after filter construction, you can see dB results on any scale of your choosing in the frequency domain displays. We have found the FIR Filter Construction function to be very reliable at building filters that have maximum gain both significantly more and less than 0 dB.

When performing inverted trace output, the current Spectrum Analyzer data display format can be of any type except for time or phase, the vertical scale can be in log or linear modes, and the current vertical scaling control ('PgUp' and 'PgDn'keys) can be at any setting.

2) The default zero dB point is 1/2 the current displayed trace, unless the crosshair cursor is displayed at the time the 'W' key is pressed. In this case, the zero dB point is the vertical position of the crosshair cursor.

3) The desired frequency response file created specifies a maximum FIR length of 769 taps. If you change this value, you should probably maintain an odd value, for the following reason: FIR filters which specify zero gain at the Nyquist frequency (half the sampling frequency), such as bandstop, highpass, and Hilbert transform, should have an odd length. It is highly likely that arbitrary responses created using the inverted filter output function will have zero or near-zero gain at the Nyquist rate also. If they do not, the odd length will not hurt anything. Reasons to decrease this value:

a. when combined with high sampling rate, 769 taps is too much for the DSP to do in real-time

b. a smoother response is required; one that outlines the general shape of the design-goal, and avoids narrow "spikes" and "dips"

4) When the inverted filter output function is enabled, the displayed magnitude trace will also be output directly to a frequency waveform file, as usual. However, it will have an "_f" attached to its waveform filename, in anticipation of the existence of a frequency response waveform file after the FIR arbitrary filter design has been performed. In this way, the original, uninverted trace is always available for comparison with the final frequency response output of the FIR filter design.

5) The Parks-McClellan FIR design for inverted filter output desired responses will likely take between 30 and 50 iterations to complete. Coupled with the long length of these filters, it may take some time to create the desired inverse filter, depending on the speed of your machine.

REAL/IMAG OUTPUT (HYPERSIGNAL-ACOUSTIC ONLY)

In Hypersignal-Acoustic, if transfer function or impulse response display is enabled, the WRITE function ('W') key will also prompt you for "real/imag output". If answered affirmatively, the program will, in addition to the standard output as described above, also create an real and imaginary waveform files suitable for inverse FFT calculation.

NOTE: **Currently the real and imaginary data written out is not affected by averaging entry. Assuming that the real and imaginary data was intended to produce an impulse response using inverse FFT Generation, then the real-time impulse response display mode in the Spectrum Analyzer itself would be a better option, since it can perform averaging, and the captured display data can be written directly to a time domain waveform file.**

4.7.3 Using Digital Record/Capture

Often times it is desirable to capture a very large amount of data at very high sampling rates in order to accomplish difficult analysis "after the fact", in non-real-time. Some examples of such situations:

1) "finding" an event that does not occur often

2) capturing the pre-event data that may have caused the event, or locating symptomatic data, and then discovering what caused it

3) studying in detail a communications and protocol sequence between different terminals, in order to resolve "we sent this, and then you sent that" type of questions

The Analog Conversion menu selection offers this type of capability. Totally designed for speed, this function offers control over sampling rate, amount of data, analog hardware parameters, and, if applicable, DSP software options such as gain, scaling, and real-time filters.

Please see section 4.1.2, Acquiring Waveforms Using DSP/Acquisition Hardware, for a complete explanation of the Analog Conversion menu selection.

4.7.4 Using "Stimulus & Response"

Often times it is desirable to test, or "excite", an unknown system with a known stimulus. By analyzing the response of the system, much can be learned about it, including frequency response, impulse response, overall delay, etc. Although the Tutorial contains complete examples of stimulus & response measurements (see chapter 12 in the Tutorial, STIMULUS & RESPONSE, and Tutorial section 15.5, Stimulus & Response Macro), some information about the actual simultaneous input/output operation in the Analog Conversion function is given below.

For DSP-based hardware only, entering a "stimulus" output waveform filename in the ACQUIRE/GENERATE field of the Analog Conversion menu causes the stimulus waveform to be sent to the D/A output at the same time an input waveform is being continuously stored to disk. Continuous storage can begin before or after onset of the stimulus, depending on the DELAY subfield in the menu. See section 4.1.2, Acquiring Waveforms From DSP/Acquisition Hardware, for detailed instructions on how to enter a stimulus waveform filename and delay.

When you invoke stimulus & response, the following sequence occurs:

1) The software preloads the stimulus into DSP board memory. *Stimulus output waveform files are never read from disk simultaneously with continuous storage of analog input.* Thus, the amount of available DSP memory takes on primary importance, since the Hypersignal software automatically sizes this memory when first initializing the DSP board. If the acquisition period is longer than the stimulus, then the stimulus repeats when the end of the stimulus is encountered, as many times as necessary. If you are trying to construct an indefinite, "seamless" stimulus, such as a pure tone, then you should build the stimulus waveform so that the length contains an exact integer number of periods. If you need zero data between stimulus repetitions (presumably to effect a delay), then the stimulus output waveform file must have zero data added to the end of the stimulus. If the stimulus is longer than all available memory, then the stimulus will be truncated and start over when the end of memory is encountered. If the acquisition period is shorter than the stimulus, then the stimulus output will also be truncated. See below for a discussion of maximum input buffer and stimulus output buffer sizes, both with standard memory size and expanded memory sizes, for different DSP chips.

2) The software instructs the DSP chip to utilize a value of "stimulus delay", taken from the DELAY subfield of the TRIGGER MODE field in the Analog Conversion menu. DELAY values must be specified in seconds. Positive DELAY values cause acquisition to begin the specified amount of time *after* the stimulus begins. Negative DELAY values cause acquisition to begin the specified amount of time *before* the stimulus begins. Note that these times must be added or subtracted to the maximum acquisition time allowed for stimulus output by the amount of memory on the DSP board. The default entry for DELAY is zero, causing the stimulus and input recording to begin simultaneously.

3) The software begins the acquisition operation. If a snap-in real-time filter has been specified, then filtering begins immediately on analog input, and remains enabled for the duration of the entire operation. Stimulus output is never filtered by the snap-in filter. If the monitor mode has been enabled, then digital loopback occurs until you answer the "BEGIN ACQUISITION ?" prompt. At this point, either the stimulus output starts (stimulus delay is positive), or acquisition starts (stimulus delay is negative), or both start simultaneously (delay is zero). When acquisition begins, trigger level and mode parameters are applied as usual. Thus, depending on trigger conditions, actual storage of data to disk might still not begin for some time, although the acquisition process has, by definition, started.

Stimulus & Response DSP Memory Usage

Below is a discussion of typical input and stimulus buffer sizes for different DSP chips, including the Texas Instruments TMS320C25 and TMS320C30, the Analog Devices ADSP-210x, the Motorola DSP56001 and DSP96002, and the NEC 77220:

TMS320C25

For TMS320C25-based DSP boards, stimulus & response operation is not enabled unless the board contains at least 16 kW (word=16 bits) of program memory. Hypersignal always loads the stimulus starting 2000H in program memory, allowing for a maximum of 8 kW of actual program information. Any expanded program memory above the standard 8 kW is detected by the software and used for stimulus output when appropriate. Input buffers for the TMS320C25 are normally calculated as the amount of data memory found on the board minus 400H. Maximum input buffer size used is 24576 (2 buffers of 12288 each). Thus, you can basically calculate the amount of memory available for stimulus output by subtracting 8 kW from the total amount of program memory on the board.

DSP32C

For DSP32C-based boards, stimulus & response operation is always enabled. Most boards come standard with at least 16 kW (word=32 bits) of total memory. Hypersignal normally uses up to 4 kW for program storage, and up to a maximum of 12 kW (two buffers of 12288 integers) for input buffer storage. If no additional memory is found by the software, then Hypersignal will *halve* the input buffer size, and use the other 6 kW for stimulus output (total of 12288 16-bit stimulus samples). If any additional memory is found, then the input buffers remain 6 kW each, and the additional memory is used for stimulus output. Thus, you can basically calculate the amount of memory available for stimulus output by subtracting 16 kW from the total amount of memory on the board (in 32-bit words), and multiplying by two (for integer stimulus sample storage).

TMS320C30

For TMS320C30-based boards, stimulus & response operation is always enabled. Most boards come standard with at least 32 kW (word=32 bits) of total memory. Hypersignal normally uses up to 8 kW for program storage, and up to a maximum of 24 kW (two buffers of 12288 words) for input buffer storage. If no additional memory is found by the software, then Hypersignal will *halve* the input buffer size, and use the remaining memory for stimulus. If any additional memory is found, then the input buffers remain 12 kW each, and the additional memory is used for stimulus output. Thus, you can basically calculate the amount of memory available for stimulus output by subtracting 32 kW from the total amount of memory on the board (in 32-bit words). If this a negative number, then divide the total amount of memory above 2000H by two.

DSP56001

For DSP56001-based boards, stimulus & response operation is always enabled. Most boards come standard with at least 48 kW (word=24 bits) of total X:, Y:, and P: memory. Hypersignal normally uses up to 4 kW of P: memory for program storage, and up to a maximum of 12 kW of X: and Y: memory each (two buffers of 12288 words), *starting at an X: and Y: memory offset of 200H,* for input buffer storage. If no additional X: and Y: memory is found by the software, then Hypersignal will *halve* the input buffer size, and use the remaining amount of combined X: and Y: memory for stimulus output. If any additional memory is found, then the input buffers remain 12 kW each, and the additional memory is used for stimulus output. Thus, you can basically calculate the amount of memory available for stimulus output by subtracting 24 kW + 200H from the total amount of X: and Y: memory on the board added together (in 24-bit words). If this a negative number, then divide (total amount of X: and Y: data memory – 400H) data memory by two.

ADSP-210x

For ADSP-210x-based boards, stimulus & response operation is always enabled. Most boards come standard with at least 8 kW (word=16 bits) of total memory. Hypersignal normally uses up to 4 kW for program storage, and up to a maximum of 12 kW (two buffers of 6144 words) for input buffer storage. If no additional memory is found by the software, then Hypersignal will *halve* the input buffer size, and use the remaining memory for stimulus output. No further memory expansion is possible for ADSP-210x boards, due to their memory addressing architecture. Thus, you can basically calculate the amount of memory available for stimulus output by taking the total amount of data memory (which is maximum of 12 kW for ADSP-210x boards) and dividing by two.

NEC77220

For 77220-based boards, stimulus & response operation is always enabled. Most boards come standard with at least 4 kW (word=32 bits) of program memory and 4 kW (word=24 bits) of data memory. Hypersignal normally uses up to 4 kW for program storage, and up to a maximum of 24 kW (two buffers of 12288 words) for input buffer storage. If no additional memory is found by the software, then Hypersignal will *halve* the input buffer size, and use the remaining memory for stimulus output. If any additional memory is found, then the input buffers remain 12 kW each, and the additional memory is used for stimulus output. Thus, you can basically calculate the amount of memory available for stimulus output by subtracting 12 kW from the total amount of data memory on the board (in 24-bit words). If this a negative number, then divide the amount of data memory by two.

DSP96002

For DSP96002-based boards, stimulus & response operation is always enabled. Most boards come standard with at least two banks of 64 kW (word=32 bits) memory. Normally, there will be at least 64 kW of memory available for stimulus output storage, and possibly more. Consult the appropriate "Onboard Memory Usage" sections for specific boards in the Hardware Reference Guide for exact amounts.

Creating Stimulus Waveforms

Again, for excellent and detailed examples of creating stimulus waveforms, see chapter 12 in the Tutorial, STIMULUS & RESPONSE. Also, Tutorial section 15.5, Stimulus & Response Macro, describes a complete, one-step macro that generates a sweep waveform, performs a stimulus & response operation, and then processes the resulting response to determine the test system's impulse and frequency responses.

4.7.4.1 Transfer Function Analysis

Transfer function analysis is commonly used during the process of characterizing a system (circuit, physical structure, etc.) when both the input and output for the system are unknown. Transfer function analysis allows one to "tap off" from any two points in a system, and determine the transfer function of the data transmission path between the two points.

Hypersignal contains two different transfer function analysis functions:

1) file-based "Transfer Function" frequency domain menu selection

2) Spectrum Analyzer real-time transfer function display (real-time software only)

Transfer Function Menu Selection

The Transfer Function menu selection takes two sets of previously created real and imaginary frequency domain waveform files (".IMA" and ".REA" filename extensions, see section 3.8.1, Creating Complex Waveform Files) and generates a new real and imaginary waveform set, along with magnitude and phase waveform files as well. The new real and imaginary data are based on the equation:

$$x+yj = a+bj \,/\, c+dj$$
$$= (ac+bd)/(cc+dd) + (bc-ad)j/(cc+dd)$$

where x and y are the output real and imaginary data, a and b are the first waveform input real and imaginary data, and c and d are the second waveform input real and imaginary data. If a+bj and c+dj represent output response and input stimulus with respect to an unknown system, then x+yj will represent the transfer function of the unknown system.

See section 4.7.4, Using "Stimulus & Response", for a discussion of this type of measurement using the Analog Conversion function.

NOTES: **1) You must have performed a previous FFT Generation operation on each input waveform with the REAL/IMAG OUTPUT menu field enabled. The FFT analysis parameters used for each waveform should be identical. The maximum length between the two waveforms is used in Transfer Function processing. If the lengths are different, then the last FFT output frame of the shorter waveform is used repetitively until the end of the longer waveform is reached.**

2) It is highly suggested that you use a no overlap and a framesize value that encompasses the stimulus length when producing the FFT Generation output waveforms that will be used as input to the Transfer Function menu selection. Segmenting the input waveforms can cause phase discontinuity problems and misleading magnitude results in the Transfer Function output. Segmentation can be valid if each FFT frame contains "independent" data; for example, a swept sine input that repeated every 4096 samples could be processed in the FFT Generation function with a framesize of 4096. Transfer Function analysis on the resulting complex frequency waveforms would then generate a series of system response frames that could be viewed in the 3-D Spectrogram or 2-D Spectrograph display functions.

3) An error message will result if the FFT lengths used for both the input waveforms are not the same. However, even though not recommended, the framesizes and windows can be different, in which case the output waveform will have the attributes of the X1 FILENAME waveform.

Please note also that the "Spectrum Analyzer" instrumentation display function contains a real-time transfer function display for selected DSP hardware.

Transfer Function Menu Fields

X1[z] FILENAME:

This field should specify the filename of the complex frequency domain waveform set containing real and imaginary data that represent unknown system "output", or as the *numerator* (a+bj) in the equation shown above. If not found, a directory of ".REA" or ".IMA" files will appear in the System Status area.

X2[z] FILENAME:

This field should specify the filename of the complex frequency domain waveform set containing real and imaginary data that represent unknown system "input", or as the *denominator* (c+dj) in the equation listed above. If not found, a directory of ".REA" or ".IMA" files will appear in the System Status area.

H[z] FILENAME:

This field should specify the filename of the complex frequency domain waveform set that will contain real and imaginary data that represent the ratio, or transfer function of the X1 FILENAME waveform divided by the X2 FILENAME waveform. Magnitude and phase output waveforms are created also to allow analysis of results using all available Hypersignal frequency domain displays.

To see transfer function results, use any of the Magnitude Display, Phase Display, 3-D Spectrogram Display, or 2-D Spectrograph Display functions. Note that you can also perform dual- or overlaid-trace display of either input or output along with the transfer function results.

Spectrum Analyzer Transfer Function Display

The Spectrum Analyzer function contains extensive transfer function and impulse response calculation and display options. A short example is given here, but you should consult section 4.7.2.2, Spectrum Analyzer Data Formats, for fully detailed information on tranfer function and impulse response operation in the Spectrum Analyzer. Also, several excellent and detailed examples are given in chapter 11 of the Tutorial, SPECTRUM ANALYZER.

Typically, transfer function display in the Spectrum Analyzer interactive display is accomplished by entering 'Hnn' in the TRACE1 or TRACE2 fields in the Spectrum Analyzer menu. "nn" refers to a frame averaging value that can range from 1 to 00 (blank entry is assumed to be 1). Also, an optional '+' or '-' entry can be added as a suffix to invoke log or negative log display. For example, a data-entry menu such as:

TRACE1 (An=anal):	a0
TRACE2 (An=anal):	a1
F1(Pnn/M+/H/T/PH):	m+
F2(Pnn/M+/H/T/PH):	h20+
LOG FREQ (N/base,NN):	n
ANALOG SAMPLING RATE:	44100
INPUT RANGE (Volts):	5
FFT ORDR, FRMSIZ, 3D FRMS:	10,256,75
WINDOW (R/Hm/Hn/Bl/Bt):	Hm
TPL (nD,D/O,LIN/PNT,S/I):	2D,D,lin,i

would specify dual-trace display from with analog channel 0 input log magnitude display on the top trace, and log magnitude transfer function (analog channel 0 complex output divided by analog channel 1 complex output) with 20-frame averaging on the bottom trace.

Spectrum Analyzer transfer function display is calculated as a complex ratio (TRACE1/TRACE2) as described by the same equation used for the Transfer Function menu selection (see above). After this calculation, magnitude, phase, and power spectra calculations are performed as usual—as if the complex data had originated directly from the FFT output. However, during Spectrum Analyzer display, log scaling and display is performed so that 0 dB appears at the center of the trace display. See section 4.7.2.2, Spectrum Analyzer Data Formats, for more information on tranfer function operation Spectrum Analyzer data-entry menu fields.

4.7.5 Using a "DSP Accelerator"

Supported DSP boards can be used "accelerate" some Hypersignal time domain functions. To select specific DSP-based hardware, set the DSP ACCELERATOR field in the System Configuration menu (see section 2.5.2).

Depending on the amount of DSP memory on individual DSP boards, acceleration may not be possible in certain cases. For example, in the FFT Generation function, if a DSP board does not have enough memory to support a 4096-pt. FFT, then a prompt will appear asking you whether to continue using the host computer 80x86 and 80x87 processors.

In the Convolution/Correlation function, relatively large amounts of DSP memory are required to accelerate the function. This is because, in order to achieve significant processing time savings, the function must read in large chunks of input waveform data for processing by the DSP board, otherwise excessive time is spent transferring points and communicating with the DSP board. Generally, most DSP boards require the manufacturer's memory expansion to activate Convolution/Correlation acceleration. Again, if the required amount of memory is not available, a prompt indicating the amount needed and asking you whether to continue using the host 80x86 processor will appear.

NOTE: At any time, for any reason, you can re-enter the System Configuration menu and disable the DSP ACCELERATOR field with a 'none' entry. When using a DSP board with a fixed-point DSP device, it may be desirable at times to disable the board and use the host 80x86 and 80x87 processors to achieve floating-point precision. An example of such a case might include performing a FFT Generation on a filter impulse response, where a precise frequency response is required.

In the FFT Generation function, the last two data-entry menu fields do not appear when a valid DSP ACCELERATOR has been specified. The PRECISION field is not active because the DSP board will use its natural precision (natural word-length of the DSP device), and the MAGNITUDE/PHASE APPROXIMATION field is not active because all DSP boards use optimized

squareroot and acrtangent functions to calculate magnitude and phase. The DSP devices are so fast that using approximations would result in inaccuracy in return for no significant processing time savings.

For information about direct modification of DSP source code in order to change the algorithms used by a DSP board when accelerating functions, or to add a different one, see section 4.7, Using DSP/Acquisition Hardware with Hypersignal.

4.8 Designing Filters In Hypersignal

Hypersignal contains extensive facilities for designing both "classical" and "arbitrary" filters. "Classical" refers to design methods based on pre-determined mathematical formulae that will produce a filter with certain characteristics and shape once the input parameters to the mathematical formula have been specified. "Arbitrary" refers to design methods that allow the user to specify the filter shape and characteristics first, then work backwards to get the necessary parameters to create such a filter. Classical methods are relatively easy to implement and can produce guaranteed characteristics such as monotonically decreasing stopband, flat passband, etc. depending on the method. Also, classical methods can tell you in advance what filter order will be required for different design techniques. Arbitrary methods are iterative and thus require computer-aided calculation—they depend on achieving some mimimum level of error in approximating the desired frequency response in order to be deemed successful. Also, the filter order (or length) cannot be predicted in advance, and in fact, *must be part of the initial specifications.* With arbitrary methods, nothing is guaranteed—not the time taken to arrive at an answer, nor the final form of the answer, although in recent years methods have been developed that have proven fairly reliable when used carefully.

Both classical and arbitrary design methods can be used to create "standard" filter types: lowpass, highpass, bandpass, and bandstop. However, only arbitrary methods can be used create "non-standard" types such as multiband, irregularly-shaped passband, curves, etc.

FIR vs. IIR Filters

Both classical and arbitrary methods can be used to create FIR and IIR filters. FIR stands for "finite impulse response" and refers to filters with transfer functions that have no "poles", only "zeros". IIR stands for "infinite impulse response" and refers to filters with transfer functions that have both poles and zeros. Since poles exist in the denominator of a transfer function, they imply feedback. Feedback, of course, is previous output being used as current input, theoretically allowing the transfer function to provide infinite output even after external input is removed. Obviously, in practice IIR filters are only useful if their transfer function does indeed "die out", or decay to zero when input is removed—if not, then the filter is said to be "unstable". FIR filters, without any poles in their transfer function, can only provide output for a finite period of time after input is removed. Generally speaking, FIR and IIR filters have the following two major tradeoffs:

1) Given enough coefficients, FIR filters can achieve any degree of "goodness" in terms of passband roll-off, ripple, stopband attenuation, etc. without becoming unstable. An IIR filter can always achieve the same level of "goodness" as its FIR counterpart with less coefficients, but can be very difficult to make stable in some cases. Since the number of coefficients translates into computation time, the choice of which type to use is often based upon how fast the filter must execute, especially for real-time applications.

2) FIR filters provide exact control over the phase of the output waveform; generally, "linear phase" FIR filters are used that provide a constant amount of delay for all frequencies (group delay) being filtered. IIR filters can do some nasty things to the phase of the filtered waveform—in some cases where group delay is very important, a second, "allpass" IIR filter must be applied that equalizes the phase distortions of the first IIR filter.

General Hypersignal Design Procedure

Hypersignal filter design always begins by entering "frequency specifications" in the FIR or IIR Filter Construction data-entry menus. From there, you are presented with a picture of the filter's "desired frequency response curve" displayed in the "graphical filter editor". If the filter design method is classical, then "tolerance regions" are displayed in the graphical filter editor—the actual filter response must fall within these tolerance regions if the filter is to meet your design specifications. If the filter design method is arbitrary, then you can further edit the desired frequency response curve using the mouse (tolerance regions can also be temporarily enabled for measurement purposes).

For both classical and arbitrary filters, the graphical filter editor contains several measurement features, such as log/linear axes, crosshair cursor readout (in dB and Hz), overlaid display with a previous actual response, grid display, etc. that are intended to aide you in the filter design process. Because pre-defined mathematical formulae are used for classical filter design, editing is

disabled and only the measurement commands in the graphical filter editor are active. After you decide that the desired frequency response shown in the graphical filter editor is appropriate, you can begin the filter construction process.

If the filter design is classical, then the appropriate mathematical sequence is applied and the filter coefficients produced relatively quickly. For classical IIR filters, the entered frequency specifications are first converted to S-plane poles and zeros, then mapped onto the Z-plane (digital) using a "bilinear transformation".

If the filter design is arbitrary, then an iterative process begins that attempts converge to a solution that provides an approximation to the desired frequency response. For arbitrary IIR filters, the design process takes place completely in the Z-domain. The iterative process for arbitrary filters can take a considerable amount of time, depending on the complexity of the desired frequency response.

NOTE: **Before continuing, something that seems so obvious to anyone familiar with filter design, but could be overlooked by someone not, should be mentioned. Any of the following design changes listed below will increase the filter's "impact" when used, which means larger order (IIR) or length (FIR), more processing time to implement, and more time for the input signal to pass through the filter (signal delay). For IIR filters, this also means greater difficulty in implementing the filter, including greater sensitivity to numerical error (mostly a fixed-point DSP chip problem), and higher "Q" (impulse response is more oscillatory). The following design changes will increase both the "difficulty rating" of implementing the filter and its impact on the input signal:**

1) increased stopband attenuation
2) decreased transition bandwidths
3) decreased ripple
4) decreased passband width (or stopband in the case of a bandstop filter)

When filter construction is complete, two output files are created:

1) The impulse response of the filter, in the form of a time domain waveform file that can be used for subsequent display, convolution, etc.

2) A filter coefficient file, in ASCII floating-point format, containing Z-plane coefficients, that can be used for subsequent recursive filtering, code generation, pole-zero display, and "snap-in" real-time filtering operations.

An FFT Generation function can then be quickly performed on the impulse response, and the Filter Construction graphical editor display re-invoked to compare the desired frequency response with the actual frequency response. Of course, the designed filter can also be tested, using the Convolution, Recursive Filtering, or "Snap-In" Real-Time Filtering functions.

Please note that there are several filter design examples in the Tutorial, including step-by-step menu set-ups and resulting waveform displays.

4.8.1 Designing Classical Filters In Hypersignal

Classical IIR filter design methods in Hypersignal include Butterworth, Chebyshev I and II, elliptic and Bessel. Classical FIR filter design methods in Hypersignal include Kaiser Window and Hamming Window. Only lowpass, highpass, bandpass, and bandstop filters can be designed using classical methods.

In the case of classical filter design, tolerance regions corresponding to the frequency specifications entered in the menu are displayed in the graphical filter editor prior to filter construction, and several measurement functions are active—but no editing is allowed. The tolerance regions are based entirely on what frequency specifications were entered in the data-entry menu—bandwidth, stopband attenuation, ripple, etc.

From the graphical filter editor, after you have decided that the displayed frequency response is correct, the CONSTRUCT FILTER command ('C' key) can be used to begin the filter construction process. Before you begin the construction process, several measurement commands are available—see section 4.3.1, Interactive Waveform Display Commands.

Classical IIR Filter Order

For classical IIR filters, the necessary filter order is pre-determined and then the correct number of S-plane poles and zeros (if any) are laid out as a lowpass filter according to the mathematical algorithm for the chosen design method. The entered frequency specifications are then "pre-warped" and used to map the S-plane transfer function into a Z-plane (digital) transfer function using an appropriate "bilinear transformation". A different bilinear transformation is used depending on whether the final filter must be lowpass, highpass, bandpass, or bandstop. Bandpass and bandstop filter orders may be as high as 40 for all classical IIR design methods except for Bessel, which is limited to 20. Lowpass and highpass filter orders may be as high as 20 for all methods except for Bessel, which is limited to 10.

Note that for arbitrary IIR filters (Steiglitz method), the filter order is entered as part of the menu specifications, instead of calculated. See section 4.8.2, Designing Arbitrary Filters in Hypersignal.

Classical FIR Filter Length

For classical FIR filters (Kaiser Window), the design takes place directly in the digital time domain. Prior to filter construction, the minimum length is estimated based on the entered frequency specifications. If the estimation does not agree with the current FILTER LENGTH menu entry, you will be prompted as to which to use. Unlike IIR filters, there is no precise calculation that relates FIR filter length to frequency response specifications. The best Hypersignal can do is provide estimated mimimum filter lengths that have been calculated from look-up table interpolation and empirically derived rule-of-thumb formulae. Generally, estimations will be within 1% to 3% of the optimal length, and are always adjusted for any odd-even length requirement (filters with non-zero gain at half-sampling frequency, such as highpass and bandstop, should be odd, Hilbert transform filters odd, and differentiators even). Passband ripple values less than 0.5 dB or larger than 3 dB are especially hard to land on exactly and may need some manual length adjustment. Kaiser Window and Hamming Window designs require an appropriate window function to be constructed that will produce the correct filter shape (lowpass, highpass, etc.) when transformed into the frequency domain. The window function internally requires a "beta" parameter, which is also estimated using the entered frequency specifications (beta is based on passband ripple, stopband attenuation, and transistion bandwidth). The maximum Kaiser Window or Hamming Window filter length is 4096.

Note that for arbitrary FIR filters, the necessary filter length is also estimated prior to filter construction and entered in the data-entry menu. As with classical filters, if the estimation does not agree with your entry, you will be prompted as to which to use. However, arbitrary filters can be edited, which can cause the required length to change dramatically from what the software has estimated. See section 4.8.2, Designing Arbitrary Filters in Hypersignal.

4.8.1.1 Using the Filter Construction Data-Entry Menus

Ths following paragraphs explain how to enter the Filter Construction menu fields in Hypersignal.

FILENAME:

This field should specify the filename for the output filter coefficient file and time domain impulse response file. FILENAME also identifies an input filter specification file to be used to initialize the menu and graphical filter editor, if one already exists.

After completion, the coefficient file may be used as input for subsequent Recursive Filtering, Pole-Zero Plot, "Snap-In" Real-Time Filtering, and Code Generate/Quantize operations, and is created with a ".FIR" extension for FIR filters, and a ".IIR" extension for IIR filters. See section 4.8.3.1, Filter Coefficient File Format.

After completion, the impulse response file may be used for subsequent display, FFT analysis (to calculate the actual filter frequency response), convolution (FIR), and recursive filtering (IIR) and other time-domain operations, and is created with a ".TIM" extension.

The specification file contains all input filter design menu parameters and the graphical desired frequency response curve for previously designed filters (see section 4.8.3.2, Filter Specification File Format). If FILENAME identifies an existing filter specification file (these files have ".FFD" filename extensions for FIR filters and ".IFD" filename extensions for IIR filters) then the menu fields and graphical desired frequency response curve will be initialized with the contents of the existing file. If no specification file exists, then the tolerance regions and desired frequency response curve will be initialized completely from what you have entered in the menu fields. For classical design methods, only tolerance region display is possible. For arbitrary design methods, the desired frequency response curve is shown initially, but tolerance region display can be toggled on and off

using the 'T' key. Also, if no specification file exists, pop-up waveform directory display of filenames with ".FFD" or ".IFD" extensions will be invoked. Consecutive use of the 'PgUp' key will then display FILENAME with any extension, then filenames with ".FIR" or ".IIR" extensions.

DESIGN METHOD:

This field selects the filter design method to be used. The following entries are possible:

B	Butterworth (IIR). Butterworth filters provide a completely flat passband, and monotonically decreasing stopband. They are sometimes used in cases where no ripple can be accepted in either the passband or stopband. After Bessel filters, Butterworth filters require greater filter order than other IIR design methods—this is the price paid for no ripple.
BES	Bessel (IIR). Bessel filters are used almost exclusively in applications that require an IIR filter with approximately linear phase. Bessel filters require excessive filter order to provide "good" magnitude response. Note that the design of Bessel filters is somewhat different than other IIR filter design methods, because the required filter order can be extraordinarily sensitive to small changes in the entered specifications. It is suggested that you start with large transition bandwidths, and then adjust other parameters as needed. Then, reduce the transition bandwidths until your specification is met or until the maximum Bessel filter order is reached (10 for lowpass/highpass, 20 for bandpass/bandstop). Also, with Bessel design, asymmetrical transition bandwidths can be achieved. See the TRANSITION BANDWIDTH1 and TRANSITION BANDWIDTH2 field descriptions below.
C1	Chebyshev I (IIR). Chebyshev I filters, like Butterworth filters, also offer a completely flat passband, but with lower filter order than Butterworth filters, because ripple is allowed in the stopband.
C2	Chebyshev II (IIR). Sometimes referred to as "inverse Chebyshev" filters, Chebyshev II filters offer opposite characteristics to Chebyshev I filters: ripple in the passband, but monotonically decreasing stopband.
DIF	Differentiator (FIR). Differentiators are intended to produce the first derivative of a time domain waveform. They are designed using a pre-defined subset of the Parks-McClellan method.
E	Elliptic (IIR). The ultimate in IIR filter design in terms of filter order. Elliptic filters always produce filters with order less than other methods given the same specification. However, this is achieved with ripple in both passband and stopband.
HM	Hamming Window (FIR). A subset of the Kaiser Window design method, Hamming Window filters can be created with a specific value of the internal "beta" parameter used in the Kaiser Window equation.
HIL	Hilbert transform (FIR). Hilbert transform filters are intended to produce a 90-degree phase lag (1/4 period delay) in a time domain waveform, regardless of the waveform frequency. They are designed using a pre-defined subset of the Parks-McClellan method.
KW	Kaiser Window (FIR). The optimum window method for FIR filter design, Kaiser Window filters produce filters that are flat in the passband and ripple in the stop band.
PM	Parks-McClellan arbitrary design (FIR). The Parks-McClellan method produces optimum FIR filters compared to other methods in terms of filter length (number of taps). Parks-McClellan filters exhibit ripple in both passband and stopband areas (in fact, they are sometimes referred to as "equiripple".)
S	Steiglitz arbitrary design (IIR). The Steiglitz method is intended to produce IIR filters of which no other IIR method is capable (irregular shapes and multiband). Steiglitz filter exhibit ripple in both passband and stopband areas. The Steiglitz method uses a least-squares error minimization technique to approximate the desired frequency response; it has been indicated in the literature that similar methods using much higher exponents (than 2) may approach the optimum for arbitrary IIR filter design.

SAMPLING FREQUENCY:

This field should specify the sampling rate (in Hertz) for the filter. Maximum sampling frequency is 1,073,709,056 Hz (1.07 GHz). SAMPLING FREQUENCY must be greater than zero.

NOTE: **With digital filters, the filter SAMPLING FREQUENCY does not have to match the actual sampling frequency of the waveform to be filtered. As long as the ratio of cutoff frequencies to sampling frequency is correct, filtering will occur as needed. However, matching values are still suggested whenever possible to ensure convenient frequency domain comparison of filters and waveforms.**

CENTER FREQUENCY:

This field should specify the center frequency (in Hertz) for the filter to be constructed.

For bandpass and bandstop filters, CENTER FREQUENCY will be the midpoint of BANDWIDTH, below. To design lowpass and highpass filters, ensure that CENTER FREQUENCY is either dc (zero) or half the sampling frequency ("Nyquist" frequency), respectively. See sections 4.8.1.2 through 4.8.1.7 below for instructions on how to use this field when creating lowpass, highpass, bandpass, bandstop, differentiator, and Hilbert transform filters.

BANDWIDTH:

This field should specify the bandwidth (width of passband area) of the filter to be constructed, in Hertz. See sections 4.8.1.2 through 4.8.1.7 below for instructions on how to use this field when creating lowpass, highpass, bandpass, bandstop, differentiator, and Hilbert transform filters.

TRANSITION BANDWIDTH1 and TRANSITION BANDWIDTH2:

This field shoud specify the width (in Hertz) of the left and right transition bands.

See sections 4.8.1.2 through 4.8.1.7 below for instructions on how to use this field when creating lowpass, highpass, bandpass, bandstop, differentiator, and Hilbert transform filters.

NOTE: **Specifying a zero value for both TRANSITION BANDWIDTHs will cancel all "undefined" regions. This setting is suggested for Steiglitz IIR arbitrary designs. See section 4.8.2.1, Using the Graphical Filter Editor, below.**

For all filter design methods, TRANSITION BANDWIDTH values have a significant effect on the order (IIR) or length (FIR) required, with smaller values causing increased order or length.

STOPBAND ATTENUATION:

This field should specify the desired stopband attenuation ("loss", "rolloff", "dB down") of the filter in dB. Stopband attenuation must be less than 100 and greater than 1. See sections 4.8.1.2 through 4.8.1.7 below for instructions on how to use this field when creating lowpass, highpass, bandpass, bandstop, differentiator, and Hilbert transform filters.

This parameter greatly affects the required order (IIR) or length (FIR) of the filter, with larger values causing increased order or length. Although sections 4.8.1.2 through 4.8.1.7 may recommend higher values, usually no more than 70 to 75 dB of stopband attenuation is usually achievable in practice.

NOTE: **Bandstop filters are specified by a *negative* STOPBAND ATTENUATION value.**

FILTER LENGTH (# OF TAPS) or STEIGLITZ FILTER ORDER:

For FIR filters, this field should specify the length of the filter (in samples). For Kaiser Window and Hamming Window filters the maximum length is 4096; for all other FIR filters (Parks-McClellan or subset) the maximum length is 769.

Before filter construction actually begins, the program will provide an estimated value of the FIR filter length based on the TRANSITION BANDWIDTH, STOPBAND ATTENUATION, and RIPPLE fields. Also, the estimated value will be odd- or even-adjusted, if necessary, depending on the type of FIR filter. If the estimated value is different then the entered length, you will be prompted as to which to use. For FIR filters only, you can enter 'AUTO' in this field to allow the program to always use its estimation. Unlike some classical methods of IIR filter design, it is not possible to calculate deterministically a mimimum length that will always specifications; rule-of-thumb formulae and empirically-determined tables must be used instead.

For IIR filters, this field should specify the filter order *for Steiglitz arbitrary filters only.* This field is ignored for all other IIR design methods. Dividing by two indicates the number of 2nd-order cascade sections (in both numerator and denominator of filter); odd values are thus rounded down. Maximum Steiglitz filter order allowed is 20. In general, the higher the filter order, the closer the possible fit to the desired frequency response. However, achieving lower minimum error at higher orders generally requires substantially more iterations and processing time to obtain convergence. Also, requesting a filter order too small to satisfy an irregular frequency response may result in failure of the process.

RIPPLE,STEIGLITZ ERROR1,STEIGLITZ ERROR2:

This field should specify the passband ripple in dB for all filter design methods, *and for the Steiglitz method only, one or two "error values" (minimum error needed for convergence).*

The way in which the RIPPLE value in this field is interpreted by the program depends on the design method chosen. In the case of classical filter design, this value specifies the amount of allowable passband ripple. Furthermore, depending on the type of classical design method chosen, this value is interpreted in slightly different, but related, ways. In the case of Butterworth, Bessel, Chebyshev II, Kaiser Window, and Hamming Window filters, there is no actual passband "ripple", since these filters are, by definition, flat in the passband. In these cases, the value is instead used to indicate the maximum attenuation, or "dB down", at the passband cutoff frequency(s). In the case of Chebyshev I, elliptic, Steiglitz, and Parks-McClellan filters, the value actually does specify the amount of true ripple allowed in the passband.

For Parks-McClellan filters, if the actual frequency response ripple fails to meet specifications, increase filter length slightly (and vice versa if actual ripple exceeds specifications). For PASSBAND RIPPLE values less than 0.5 dB or greater than 3 dB, it may prove desirable to override estimated filter length with increased values to achieve less ripple and decreased values to achieve greater ripple. The same sort of tradeoff is basically true for Steiglitz filters: increasing the ripple value may require a lower filter order (see STEIGLITZ FILTER ORDER field, above), or make convergence easier.

Steiglitz Error Values

In the case of Steiglitz arbitrary filter design, this field accepts two additional values. Both values represent the minimum error necessary to obtain convergence, with the first value being used for "initial convergence", and the second for convergence thereafter. This two-level technique is used because more often than not the Steiglitz method must invert unstable poles and zeros outside the Z-domain unit circle after initial convergence, which then results in lower error values for the next convergence. Thus, two-tier entry can avoid an endless search for first convergence that meets a very low minimum error, and instead apply the low minimum error only after a basic pole-zero pattern has been established. If only the STEIGLITZ ERROR1 value is entered, then the STEIGLITZ ERROR2 value is assumed to be the same. A typical entry example might be:

'RIPPLE,1e-3,1e-7'

To achieve a "preview" of a Steiglitz filter that might ordinarily required hundreds of iterations in 50-75 iterations, enter STEIGLITZ ERROR1 only, at a value of about .001. To achieve a "good" filter, choose STEIGLITZ ERROR2 to be at least 1E-7. It is suggested that after obtaining convergence with a "stiff" ERROR2 value, that you try decreasing the ERROR1 value also, as having both values low can sometimes force the Steiglitz method to search for different convergence patterns initially (which may or may not yield a good result). As you may have gathered, there is no sure thing with the Steiglitz method (and, in fact, not with any IIR filter arbitrary method that we know of). You might have to experiment with the STEIGLITZ ERROR1 and STEIGLITZ ERROR2 values.

After at least 2x FILTER ORDER iterations, the current value of error is displayed continuously on the menu throughout the design process. Although the current error can "jump up" frequently as new minimums are found, it remains a good estimate of the likelihood of convergence, even though it is possible that convergence may be obtained through other criteria. Decreasing the minimum error required generally improves the resulting frequency response approximation.

4.8.1.2 Designing Lowpass Filters

This section assumes that you are familiar with the Filter Construction menu data-entry fields (section 4.8.1.1 above, Using the Filter Construction Data-Entry Menus).

To design a lowpass filter, set the CENTER FREQUENCY field to zero. This will cause the program to interpret the value of the BANDWIDTH field as the filter "cutoff" frequency (width of the filter's passband). Set the TRANSITION BANDWIDTH2 field as required by the roll-off of the filter; the TRANSITION BANDWIDTH1 field is ignored.

4.8.1.3 Designing Highpass Filters

This section assumes that you are familiar with the Filter Construction menu data-entry fields (section 4.8.1.1 above, Using the Filter Construction Data-Entry Menus).

To design a lowpass filter, set the CENTER FREQUENCY field to the "Nyquist frequency": *half the value of the* SAMPLING FREQUENCY *field.* This will cause the program to interpret the value of the BANDWIDTH field as the distance from the Nyquist frequency to the filter "cuton" frequency (width of the filter's passband). Set the TRANSITION BANDWIDTH1 field as required by the roll-off of the filter; the TRANSITION BANDWIDTH2 field is ignored.

NOTE: For FIR highpass filters, the program automatically adjusts length estimates to be odd values. This is because FIR filters with even symmetry that have a high magnitude value at the Nyquist frequency are required to have an odd length value (which produces the appropriate Nyquist frequency response about n=0). If you override the length estimate, please continue to use an odd value for highpass filters.

4.8.1.4 Designing Bandpass Filters

This section assumes that you are familiar with the Filter Construction menu data-entry fields (section 4.8.1.1 above, Using the Filter Construction Data-Entry Menus).

To design a bandpass filter, set the CENTER FREQUENCY field to the midpoint of the bandwidth of the filter, and set the BANDWIDTH field to the bandwidth (passband) of the filter. Set the TRANSITION BANDWIDTH1 and TRANSITION BANDWIDTH2 fields as required by the roll-off of the filter. The minimum of the transition bandwidth fields is used to determine the filter order or length. Since classical designs always produce symmetrical filters, the minimum transition bandwidth will control both transition bandwidths. Arbitrary designs can be used to achieve asymmetrical transition bandwidths.

4.8.1.5 Designing Bandstop Filters

This section assumes that you are familiar with the Filter Construction menu data-entry fields (section 4.8.1.1 above, Using the Filter Construction Data-Entry Menus).

To design a bandstop filter instead of a bandpass filter, set the STOPBAND ATTENUATION field to a *negative value*. In this case, the BANDWIDTH field is interpreted as the width of the filter's stopband. Otherwise, there is no difference between bandstop and bandpass filter specification.

NOTE: For FIR bandstop filters, the program automatically adjusts length estimates to be odd values. This is because FIR filters with even symmetry that have a high magnitude value at the Nyquist frequency should have an odd length value (which produces the appropriate Nyquist frequency response about n=0). If you override the length estimate, please continue to use an odd value for bandstop filters.

4.8.1.6 Designing Differentiation Filters

This section assumes that you are familiar with the Filter Construction menu data-entry fields (section 4.8.1.1 above, Using the Filter Construction Data-Entry Menus). Although this section is a subsection of "Designing Classical Filters", the differentiator selection in Hypersignal actually invokes a pre-defined subset of the Parks-McClellan arbitrary filter design method.

To design a differentiator, set both the CENTER FREQUENCY and TRANSITION BANDWIDTH1 fields to the "Nyquist frequency": *half the value of the* SAMPLING FREQUENCY *field.* Set the BANDWIDTH field to zero; the TRANSITION BANDWIDTH2 field will be ignored. Set the STOPBAND ATTENUATION to 80 dB or greater, and the RIPPLE field as required.

Differentiators are intended to produce the first derivative of a time domain waveform. As per the guidelines above, their desired frequency response should be linear with respect to frequency.

NOTE: **For FIR differentiator filters, the program automatically adjusts length estimates to be even values. This is because FIR filters with odd symmetry that have a high magnitude value at the Nyquist frequency should have an even length value (which produces the appropriate Nyquist frequency response about n=0). If you override the length estimate, please continue to use an even value for differentiator filters.**

4.8.1.7 Designing Hilbert Transform Filters

This section assumes that you are familiar with the Filter Construction menu data-entry fields (section 4.8.1.1 above, Using the Filter Construction Data-Entry Menus). Although this section is a subsection of "Designing Classical Filters", the Hilbert transform selection in Hypersignal actually invokes a pre-defined subset of the Parks-McClellan arbitrary filter design method.

To design a Hilbert transform filter, set the CENTER FREQUENCY field to one-fourth the value of the SAMPLING FREQUENCY field. Set the BANDWIDTH field to be 90% to 95% of the "Nyquist frequency" (half the sampling frequency). Set the TRANSITION BANDWIDTH1 and TRANSITION BANDWIDTH2 fields to take up the remaining distance on either side of the bandwidth (extend to zero on the left, and to the Nyquist frequency on the right). Set the STOPBAND ATTENUATION to 80 dB or greater, and the RIPPLE field as required.

The percentage of the Nyquist frequency used for the BANDWIDTH field depends on how "good" you want to make the Hilbert transform versus the length you are willing to tolerate. A greater percentage moves the resulting output filter closer to the ideals of 90-degree phase shift and complete "all-pass" magnitude response, but requires more taps.

In some cases, improved Hilbert transform designs may be attainable by setting TRANSITION BANDWIDTH values to zero and manually controlling the filter length.

Hilbert transform filters are intended to produce a 90-degree phase lag (1/4 period delay) in a time domain waveform, regardless of the waveform frequency. As per the guidelines above, their desired frequency response should look like an "all-pass", or very wide bandpass.

NOTE: **For FIR Hilbert transform filters, the program automatically adjusts length estimates to be odd values. This is because FIR filters with odd symmetry that have a low magnitude value at the Nyquist frequency should have an odd length value (which produces the appropriate Nyquist frequency response about n=0). If you override the length estimate, please continue to use an odd value for Hilbert transform filters.**

4.8.2 Designing Arbitrary Filters In Hypersignal

Hypersignal utilizes the Parks-McClellan method to design arbitrary FIR filters, and the Steiglitz method to design arbitrary IIR filters. Of course, standard designs (lowpass, bandpass, etc.) are a simple subset of arbitrary filters, and can also be designed using these methods. The Parks-McClellan method also includes differentiator and Hilbert transform designs as a subset.

In the case of arbitrary filter design, the desired frequency response may be totally arbitrary. If an irregularly-shaped passband or multiband filter is desired, you can graphically edit the desired frequency response using the mouse, or specify the desired response numerically inside a filter specification file.

NOTE: **Although we have repeatedly mentioned the ability to edit arbitrary filter designs using the mouse, we actually recommend creating specification files, which are in ASCII format, outside of Hypersignal—preferably by using an external program to generate the desired response curve specification numerically. You can also use a text-editor to create and/or modify filter specification files. The desired frequency response curve portion of these files are read (if pre-existing) and saved by the graphical filter editor whenever you work with arbitrary design methods. See sections 4.8.3, Specifying Filters Outside of Hypersignal, and 4.8.3.2 Filter Specification File Format, for much greater detail on how to create arbitrary filter designs using specification files.**

Steiglitz Arbitrary Design

For Steiglitz arbitrary IIR filters, the necessary filter order cannot be determined in advance, and is thus entered in the IIR Filter Construction menu (latter portion of the RIPPLE,STEIGLITZ ERROR1,STEIGLITZ ERROR2 field). The minimum error that must be reached for the iterative process to achieve convergence is also entered in the menu. If, after convergence, there remain unstable poles (or zeros) outside the Z-plane unit circle, Steiglitz filter design dictates that the unstable values be inverted and the resulting coefficients be used as a starting point for another iteration sequence to obtain convergence. This may happen not at all or multiple times, depending on the desired frequency response and filter order. After the design process, which operates completely in the Z-domain and thus does not require a bilinear transformation, is complete, the resulting filter coefficients are stored in an IIR filter coefficient file. The maximum Steiglitz filter order is 20.

Some further comments about the Steiglitz design process are appropriate. The desired response curve for a Steiglitz IIR filter is composed of two fundamental parts: the outline, or amplitude, of the desired frequency response, and the placement of discrete frequency points. The latter is especially noteworthy: frequency values do not have to be spaced linearly; in fact, increasing their density in amplitude transition areas can improve the resulting filter response. To allow control over these two processes, the Hypersignal graphical filter editor has two basic modes—point mode (amplitude shaping) and region mode (frequency point density control). For most Steiglitz designs, large values of STOPBAND ATTENUATION (greater than 50 dB) and transition bandwidth values of zero are suggested. Zero transition bandwidth values cause maximum density to be concentrated around the passband edges, forming a “brick wall” design goal. This will also cancel all “undefined” regions—see section 4.8.2.1, Using the Graphical Filter Editor, below.

With the current values of STEIGLITZ ERROR, STEIGLITZ FILTER ORDER, and STOPBAND ATTENUATION, the Steiglitz method then employs numerical techniques to minimize the mean-squared amplitude error evenly among all frequency points. The process *does not* guarantee a frequency response that is a specified attenuation at a certain cutoff value, but *does* allow arbitrary design and can produce near-elliptic-quality magnitude response, depending on point density and placement, and the minimum error allowed.

Parks-McClellan Arbitrary Design

For Parks-McClellan arbitrary FIR filters, the necessary filter length is estimated prior to filter construction and also entered in the data-entry menu. If the estimation does not agree with your entry, you will be prompted as to which to use. For non-standard Parks-McClellan arbitrary FIR filters, the filter length estimation may not be accurate. Normally, for standard designs (lowpass, bandpass, etc.), the area with the largest amplitude swing in the smallest frequency width (transition bandwidth), along with passband ripple, forms the basis for length estimation. If editing has modified this area, or basically modified the desired response so it doesn’t look like a standard filter anymore, then Hypersignal’s estimation may be substantially off the mark. You may have to iterate through the design process a few times to arrive at a filter length value that just meets the specifications. Also, irregularly-shaped passbands may require much-relaxed passband ripple constraints (3 dB to 10 dB) in order to meet stopband specifications. It is suggested that length be varied first until no further improvement can be obtained in the stopband, then passband ripple be adjusted until the specifications are met. The maximum filter length for arbitrary Parks-McClellan designs (and also for differentiator and Hilbert transform filters, which are a subset of the Parks-McClellan method) is 769.

Some further comments about the Parks-McClellan design process are appropriate. First, the process is iterative and the amount of time required is generally unpredictable, ranging from minutes to hours; second the process also depends, obviously, on machine speed. However, time required is definitely related to filter length, with longer filters taking more time. Three numbers are displayed to indicate progress: iterations (ITER), deviation (DEV), and rate of closure (CLOSURE). Deviation values higher than 0.1 usually mean more difficult-to-achieve filters that require more time. The process is approaching completion as closure rate, which decreases more or less monotonically, approaches zero. Usually 12-17 iterations are required, but some bandstop and arbitrary response filters may take up to 35 to 50 iterations. The Parks-McClellan design method does not guarantee a frequency response that meets all specified attenuation values, but it does *produce the optimum possible FIR design*. Even for standard filter types, the Parks-McClellan method is widely used, because it alwyas produces filters that require fewer taps (less length) than other methods.

4.8.2.1 Using the Graphical Filter Editor

Prior to all filter construction in Hypersignal, the proposed frequency response is displayed in the graphical filter editor, and various measurement commands are made active. From this display, you make the actual decision to begin constructing the filter. Note that unlike classical filter designs, the tolerance regions are not displayed initially. The actual desired shape of the fre-

quency response is displayed instead, since it can actually be edited. If, at any time during the design process, you wish to measure the actual frequency response against tolerance regions, you can temporarily display tolerance regions by using the 'T' key inside the graphical display (but you must switch back, using the 'T' key again, to continue editing).

The Hypersignal graphical filter editor allows display of desired frequency responses for both classical and arbitrary filters and comparison against actual frequency responses. Also, several measurement commands are available, including crosshair cursor readout in linear or dB amplitude and frequency (Hz). For arbitrary filter designs, you can further edit the desired frequency response with the mouse. However, for classical filter designs, no additional frequency response editing is possible, because the classical designs, by definition, have a mathematically pre-determined characteristic shape and cannot be altered.

NOTE: **The following information is very important when editing arbitrary designs: when first specified from the data-entry menu, transition bandwidths are always initialized to be "undefined regions". This is done to achieve optimal results for standard lowpass, highpass, bandpass, and bandstop filter shapes when an arbitrary filter design method has been chosen. In undefined regions, *the graphical filter editor makes no assumptions about shape*, instead allowing the arbitrary methods to do "what comes naturally", and concentrate all error distribution in the passband and stopband regions. This works extremely well for the Parks-McClellan FIR method, *unless your design uses non-symmetrical transition bandwidths, or multiple bands, in which case you should probably define all regions*. Otherwise, the resulting filter curve can meet specifications in the defined regions, but not behave well in the undefined regions, much like a higher-order polynomial can fit a set of points, but not provide good interpolation between points.**

For the Steiglitz IIR method, it is recommended that you initialize the transition bandwidth fields in the data-entry menu to zero, which will eliminate undefined regions. This is because the Steiglitz method often requires a high density of desired response points in transition areas. Of course, for both methods you may at any time "define" (or undefine, or re-define) regions in the graphical filter editor. Thus, if your arbitrary design starts out as a basic lowpass, bandpass, etc. filter, you may then "define" all transition areas, and subsequently edit the entire desired frequency response in any way necessary.

Some Suggestions For Arbitrary Filter Editing

1) First of all, if you are quite sure of your desired frequency response, then don't edit at all—write an external program that will numerically generate the required desired response curve and store it in a filter specification file that the arbitrary design methods can use as input. Any programming language will do; the resulting ".COM" or ".EXE" program can even be run directly from inside Hypersignal, if desired (see section 3.6). See sections 4.8.3, Specifying Filters Outside of Hypersignal, and 4.8.3.2 Filter Specification File Format, for much greater detail on how to create/modify desired frequency response curves in specification files.

2) When you are using the mouse to create angled lines, it is suggested that you first decrease the "density" of points that you are working with, shape the line, then re-insert points to restore the density. To control density, you must be in "region mode", which displays a target window controlled by the up and down arrow keys (or mouse); you can then use the 'Insert' and 'Delete' keys to insert and delete points. *Working with fewer points is much easier* when using the mouse to edit filter shapes. Note that after restoring increased density to newly shaped regions while in log-display mode, you may have to "touch-up" the regions with some more editing, due to the fact that Hypersignal inserts points based on linear-interpolation.

3) For exact mouse editing, rubberband edit one point at a time! Attach the cursor ('Home' key, or left button on mouse) to desired response point, position it, then detach the cursor ('Home' key or left button again). Then move to the next point, and repeat. This will avoid accidental movement of points that have already been edited.

4) Always establish greatest density in "transition areas", or regions with curves, sloped-lines, etc. In the case of the Steiglitz IIR arbitrary method this is vital, because the inner-workings of the Steiglitz algorithm are based on non-linear spacing of the desired frequency-response points. Areas with highest concentration of desired points will receive the greatest amount of "effort" by the Steiglitz algorithm—flat areas such as passband and stopband are easier for the algorithm to achieve, and thus do not require a high density of desired points. In the case of the Parks-McClellan FIR arbitrary method, increased density in shaped areas will only make slight differences.

5) To get fine control over your editing, use the 'PgUp' key to "zoom vertically" on the desired response; each incremental mouse (or arrow key) movement then covers less vertical distance.

6) Leave “undefined regions” as is (unedited) for standard (lowpass, bandpass, etc.) designs implemented with the Parks-McClellan method. However, more than two or three undefined-regions may begin to cause problems. Normally, an undefined-region works well with the Parks-McClellan method because the actual response is unconstrained in the undefined area, allowing all error-minimization “effort” by the algorithm to take place in defined areas. However, as the desired-response becomes more complicated, the actual response may begin to “overreact” in an undefined area, much the way a higher-order polynomial fit to a curve can behave wildly *between* points. You will have to experiment with complicated designs to determine when behavior in undefined areas is acceptable, and when the area should be constrained by defining at least a few desired frequency response points.

7) Eliminate “undefined regions” for standard (lowpass, bandpass, etc.) designs implemented with the Steiglitz method. Essentially, this can be done by setting desired transition bandwidths to zero in the Filter Construction data-entry menu. See 4), above.

8) When editing filters with gain > 1 (gain > 0 dB), linear display mode is suggested. By pressing the ‘PgDn’ key, you can see gain values greater than 1, and can also edit these values.

Below is a very brief listing of all commands available in the graphical filter editor; however, a more detailed listing is presented in section 4.3.1, Interactive Waveform Display Commands.

ATTACH/DETACH CURSOR: ‘Home’ key or left mouse button

Must be in point mode. The crosshair cursor is “attached” to the frequency response; subsequent movement of the cursor “drags” waveform samples along with the cursor. If the cursor is already attached, then the cursor will be detached.

CONSTRUCT FILTER: ‘C’ key

This command saves the current filter specifications, including all menu parameters, and in the case of arbitrary filter designs, the “desired response curve”, and begins the filter construction process.

CROSSHAIR CURSOR: arrow keys or mouse motion

Moves target window (region mode) or crosshair cursor (point mode).

DEFINE REGION:

See UNDEFINE REGION command.

DELETE: ‘Delete’ key

In region mode only, this command can be used to halve the density of the desired frequency response axis points in the region indicated by the target window.

ESCAPE: ‘Esc’ key or leftmost two mouse buttons on 3-button mouse

Pressing the ‘Esc’ (escape) key at any time will return to the previous data-entry menu, and the data-entry editor will be re-activated.

FILE INITIALIZE:

See MENU INITIALIZE command.

GRID: ‘G’ key

Rotates the graphical filter design grid between none, solid, intersected.

HELP: ‘?’ or ‘F1’ keys, rightmost two mouse buttons on 3-button mouse

Displays on-line Help pertaining to the current waveform display function. Use the ‘Esc’ key again to return to the current graphical display.

INSERT: **'Insert' key**

In region mode only, this command can be used to double the density of the desired frequency response axis points in the region indicated by the target window.

LOG/LINEAR: **'L' key**

This command toggles both the amplitude and frequency axes between linear and log. Current axes format applies to both desired and actual (if enabled) filter responses. Four total combinations are possible. Linear amplitude (vertical axis) display may prove useful for working with frequency responses (both desired and actual) that have maximum gain greater than 1 (zero dB). @COMMAND_LEFT = MENU or FILE INITIALIZE:

'M' or 'F' keys

Reinitialize the graphical filter display from the data-entry editor menu or from the specification file. Filter specification files have ".FFD" filename extensions for FIR filters and ".IFD" filename extensions for IIR filters, and contain both menu parameters and desired frequency response curve. All new changes to the current edited response are completely lost when either command is executed.

OVERLAY: **'O' key**

This keyboard command allows the desired response to be overlaid with the previous actual filter response, or to view either alone. Press the 'O' key three times to see all possibilities.

POINT MODE: **'P' key**

Puts the graphical filter editor into point mode. In point mode, the arrow keys or mouse motion controls a "crosshair cursor" which can be used to provide amplitude and frequency readout. For arbitrary filter design methods, frequency points can be rubberband-edited in point mode (see ATTACH/DETACH CURSOR command).

PRINT: **'F2' key**

Invokes the pop-up print menu, which allows both screen dumps and publication-quality prints of the current waveform display to nearly 200 different types of printers. The output device (LPT1, COM1, disk filename, etc.), dimensions, dots-per-inch, orientation, background, etc. can be specified in the pop-up print menu. Publication-quality prints require adequate EMS memory. Please see section 4.3.2, Printing Waveform Displays, for more detailed information.

REDRAW: **'Enter' key**

Redraws the current display using initial settings. All vertical scaling settings are reset. If in point mode, the crosshair cursor is detached from the desired response curve if attached.

REGION MODE: **'R' key**

Puts the graphical filter editor into region mode. In region mode, the arrow keys or mouse motion controls a "target window". For arbitrary design methods, desired frequency response points can be inserted and deleted in region mode.

SCALING:

See VERTICAL SCALING command.

TARGET WINDOW: **arrow keys or mouse motion**

Moves target window (region mode) or crosshair cursor (point mode).

TOLERANCE REGION ENABLE: **'T' key**

This command toggles graphical filter design tolerance region display on and off. Tolerance regions are always enabled upon entrance to graphical filter display for classical design methods, and disabled upon entrance to display for arbitrary design methods.

This command is active only in the graphical filter editor display.

UNDEFINE or DEFINE REGION: **'U' or 'D' keys**

Undefines or defines a region currently marked by the target window (must be in region mode). Undefined regions are areas where the arbitrary design is purposefully left unspecified to the design process (to allow natural convergence in that region). No desired response curve is shown on the display in undefined regions. Defining a region simply reinstates the desired response curve in that region, whether it be what the program initialized to or after user-editing.

VERTICAL SCALING: **'PgUp' and 'PgDn' keys**

The 'PgUp' and 'PgDn' keys can be used to control vertical amplitude scaling. They are active independent of other waveform display settings such as log/linear amplitude, overlay, grid, etc.

To scale a waveform up vertically (make it larger), use the 'PgUp' key. To scale a waveform down vertically (make it smaller), use the 'PgDn' key. Note that vertical amplitude units change along with waveform size.

WINDOW:

See TARGET WINDOW command.

4.8.3 Specifying Filters Outside of Hypersignal

There are two ways to specify FIR or IIR filters outside of Hypersignal:

1) Create a new (or modify an existing) FIR or IIR filter coefficient file directly using a text-editor or external program. Filter coefficient files are in ASCII format, and allow you to specify Z-plane numerator coefficients and denominator coeffcients (of the filter's transfer function) using floating-point values.

2) Create a new (or modify an existing) FIR or IIR filter specification file using a text-editor or external program. Filter specification files are also in ASCII floating-point format—after specifying a desired frequency response, you can re-execute Hypersignal and allow the Parks-McClellan FIR or Steiglitz IIR arbitrary design methods attempt to create filters with actual frequency responses that approximate your desired response. As opposed to the first method, you are specifying *the desired frequency response* of the filter, and then allowing Hypersignal to create the filter coefficient file based on this input.

Filter specification files have ".FFD" filename extensions for FIR filters, and ".IFD" filename extensions for IIR filters. See section 4.8.3.2 below, Filter Specification File Format, for a detailed description of their format.

4.8.3.1 Filter Coefficient File Format

Below are examples of actual ".IIR" and ".FIR" filter coefficient files created with the IIR and FIR Filter Construction functions. This is the format that is expected by all other Hypersignal functions that use filter coefficient files as input (Recursive Filtering, "Snap-In" Real-Time Filtering, Pole-Zero Plot, and Code Generate/Quantize). The following is an example of a ".IIR" filter coefficient file created by the IIR Filter Construction function:

```
BP
S3
259
1.0000000000E+04
FILENAME: ELLIPTIC.IIR
DESIGN: E
SAMPLING FREQUENCY (Hz): 10000
CENTER FREQUENCY (Hz): 2500
BANDWIDTH (Hz): 850
TRANSITION BANDWIDTH1 (Hz): 70
TRANSITION BANDWIDTH2 (Hz): 70
STOPBAND ATTENUATION (dB): 24
```

FIR LENGTH/IIR ORDER: 8
RIPPLE (dB),STEIGLITZ ERR: 2
(blank or comment line—cannot be removed)
FILTER TYPE (LP/HP/BP/BS): BP
(blank or comment line—cannot be removed
CONSTANT MULTIPLIER C:
4.1954311346E-02
NUMBER OF NUMERATOR (ZERO) FACTORS:
4
ORDER OF ZERO FACTOR (SECTION) 1:
2
A[0,1] (z** -0)= (comment line: text or blank)
1.0000000000E+00
A[1,1] (z** -1)=
-1.0978389470E+00
A[2,1] (z** -2)=
1.0000000000E+00
ORDER OF ZERO FACTOR (SECTION) 2:
2
A[0,2] (z** -0)=
1.0000000000E+00
A[1,2] (z** -1)=
1.0978388693E+00
A[2,2] (z** -2)=
1.0000000000E+00
ORDER OF ZERO FACTOR (SECTION) 3:
2
A[0,3] (z** -0)=
1.0000000000E+00
A[1,3] (z** -1)=
-6.3550988415E-01
A[2,3] (z** -2)=
1.0000000000E+00
ORDER OF ZERO FACTOR (SECTION) 4:
2
A[0,4] (z** -0)=
1.0000000000E+00
A[1,4] (z** -1)=
6.3550978425E-01
A[2,4] (z** -2)=
1.0000000000E+00
(blank or comment line—cannot be removed)
NUMBER OF DENOMINATOR (POLE) FACTORS:
4
ORDER OF POLE FACTOR (SECTION) 1:
2
B[0,1] (z** -0)= (comment line: text or blank)
1.0000000000E+00
B[1,1] (z** -1)=
-5.1237035873E-01
B[2,1] (z** -2)=
9.7160823126E-01
ORDER OF POLE FACTOR (SECTION) 2:
2
B[0,2] (z** -0)=
1.0000000000E+00

B[1,2] (z** -1)=
5.1237025648E-01
B[2,2] (z** -2)=
9.7160823086E-01
ORDER OF POLE FACTOR (SECTION) 3:
2
B[0,3] (z** -0)=
1.0000000000E+00
B[1,3] (z** -1)=
-2.7017721076E-01
B[2,3] (z** -2)=
8.5453421186E-01
ORDER OF POLE FACTOR (SECTION) 4:
2
B[0,4] (z** -0)=
1.0000000000E+00
B[1,4] (z** -1)=
2.7017710974E-01
B[2,4] (z** -2)=
8.5453421077E-01

NOTE: **IIR filter coefficient files, when created by the Hypersignal IIR Filter Construction function, contain 2nd-order numerator and denominator cascade sections ordered in ascending Q-order, according to the L.B. Jackson Q-Ordering Convention. Numerator and denominator sections are ordered independently; the "Q" is calculated as $|A_{i2}|/A_{i1}^2$, where A_{i1} is the first numerator coefficient of the ith section.**

The following is an example of an actual ".FIR" filter coefficient file created with the FIR Filter Construction function:

BP
32
8.0000000000E+03
FILENAME: kaiser.FIR
DESIGN: kw
SAMPLING FREQUENCY (Hz): 8000
CENTER FREQUENCY (Hz): 2000
BANDWIDTH (Hz): 1000
TRANSITION BANDWIDTH1 (Hz): 250
TRANSITION BANDWIDTH2 (Hz): 250
STOPBAND ATTENUATION (dB): 10
FIR LENGTH/IIR ORDER: 32
RIPPLE (dB),STEIGLITZ ERR: 2
(blank or comment line—cannot be removed)
FILTER TYPE (LP/HP/BP/BS): BP
(blank or comment line—cannot be removed)
CONSTANT MULTIPLIER C:
1.0000000000E+00
NUMBER OF NUMERATOR (ZERO) FACTORS:
1
ORDER OF ZERO FACTOR (SECTION) 1:
32
A[0] (z** -0)= (comment line: any text or blank)
2.8172130081E-02
A[1] (z** -1)=
-2.3003211735E-02
A[2] (z** -2)=

-1.1233741794E-02
A[3] (z** -3)=
-5.2840136758E-03
A[4] (z** -4)=
-2.3318051801E-02
A[5] (z** -5)=
3.8755954208E-02
A[6] (z** -6)=
4.7327998906E-02
A[7] (z** -7)=
-4.5425193953E-02
A[8] (z** -8)=
-3.0857127287E-02
A[9] (z** -9)=
3.3983280278E-03
A[10] (z** -10)=
-3.4993947317E-02
A[11] (z** -11)=
8.0348898526E-02
A[12] (z** -12)=
1.2722451525E-01
A[13] (z** -13)=
-1.6953754440E-01
A[14] (z** -14)=
-2.0153859961E-01
A[15] (z** -15)=
2.1875915763E-01
A[16] (z** -16)=
2.1875915763E-01
A[17] (z** -17)=
-2.0153859961E-01
A[18] (z** -18)=
-1.6953754440E-01
A[19] (z** -19)=
1.2722451525E-01
A[20] (z** -20)=
8.0348898526E-02
A[21] (z** -21)=
-3.4993947317E-02
A[22] (z** -22)=
3.3983280278E-03
A[23] (z** -23)=
-3.0857127287E-02
A[24] (z** -24)=
-4.5425193953E-02
A[25] (z** -25)=
4.7327998906E-02
A[26] (z** -26)=
3.8755954208E-02
A[27] (z** -27)=
-2.3318051801E-02
A[28] (z** -28)=
-5.2840136758E-03
A[29] (z** -29)=
-1.1233741794E-02
A[30] (z** -30)=

-2.3003211735E-02
A[31] (z** -31)=
2.8172130081E-02
(blank or comment line—cannot be removed)
NUMBER OF DENOMINATOR (POLE) FACTORS:
0

The first four lines of filter coefficient files are used by Hypersignal functions such as Recursive Filtering, Code Generate/Quantize, "Snap-In" Real-Time Filtering, and Pole-Zero Plot for initialization purposes. The first line is used to initialize Pole-Zero inverse bilinear transformation, the second line to initialize the Code Generate/Quantize IIR cascade structure field, the third line to initialize the constant multiplier in all three functions, and the fourth line to initialize the impulse response length for Code Generate/Quantize.

If you attempt to generate or modify filter coefficient files via an external program or text-editor, *you must maintain these first four lines,* or run the risk of unpredictable Hypersignal program operation. The next 14 lines represent an "information-only" storage of filter design data entry menu specifications intended to aide with technical notekeeping and project design documentation. Although these lines are not actually used as program input, they still should not be tampered with, if possible.

The remainder of coefficient files contains comments and values pertaining to the actual filter transfer function itself, and is explained below.

NOTE: ***For IIR coefficient files only,*** **some terms in the file format description below refer to the following transfer function equation.** ***The transfer function does not apply in the case of an FIR filter coefficient file:***

$$H(z) = C \frac{\prod_{j=1}^{(N+m-1)/m} A_{0j} + A_{1j}z^{-1} + A_{2j}z^{-2} + \ldots + A_{mj}z^{-m}}{\prod_{j=1}^{(N+n-1)/n} 1 + B_{1j}z^{-1} + B_{2j}z^{-2} + \ldots + B_{nj}z^{-n}}$$

where C is the constant multiplier, m and n are H(z) transfer function numerator and denominator factor orders, respectively, and N is the overall filter order.

Detailed Line Description

lines 1-18: Comment lines—discarded after being read, except for the first four lines, which are reserved for program use by other functions. The next 14 lines include information about the filter (frequency specifications, design method, filter order, etc.), and are not used as input by any Hypersignal functions. Line 1, S-plane filter type (BP, LP, HP, BS), which is used by the Pole-Zero Plotting function, line 2, filter structure (S1, S2, S3), which is used by Code Generate/Quantize and real-time filtering, and lines 3-4, truncated impulse response length (20-4096), and sampling frequency (1 - 1,073,709,056), which are used by Code Generate/Quantize. The values on these lines should *never be deleted, exceed the range, or set to values other than those just listed.*

line 19: H(z) numerator constant multiplier C (floating-point format—see below for numeric formats). For FIR filters this value should be 1.0 unless the filter impulse response is antisymmetric (e.g. Hilbert transform), in which case this value should be –1.0 (negative one).

line 20: comment line *(all comment lines are discarded after being read)*

line 21: number of numerator (zero) factors (integer format—see below for numeric formats)

line 22: comment line

line 23: order of first numerator factor (integer format)

line 24:	comment line
line 25:	first Ai coefficient (floating-point format)
line 26:	comment line
line 27:	second Ai coefficient (floating-point format)
.	
.	
.	
line 24+2∗N:	comment line
line 25+2∗N:	Nth Ai coefficient, where N is line 23 value. The sequence in lines 22 thru 25+2∗N is repeated as necessary to make the total numerator factors equal to the value in line 21.
line p:	blank line
line p+1:	comment line
line p+2:	number of denominator (pole) factors (integer format—see below for numeric formats). *For FIR filters, set equal to zero and omit remainder of format.*
line p+3:	comment line
line p+4:	order of first denominator factor (integer format)
line p+5:	comment line
line p+6:	first Bi coefficient (floating-point format)
line p+7:	comment line
line p+8:	second Bi coefficient (floating-point format)
.	
.	
.	
line p+5+2∗N:	comment line
line p+6+2∗N:	Nth Bi coefficient, where N is line p+4 value. The sequence in lines p+3 thru p+6+2∗N is now repeated as needed to make the total denominator factors equal to the value in line p+2.

NOTE: **For IIR filters, even though not actually used, *the first Bij coefficient for each factor, B0j, cannot be skipped, and should be* set equal to one. If numerator factors=1 then line 21 should equal one (1), line 23 should equal "N" in the above transfer function equation, and the zero coefficients are thus in fully expanded form. *This is essentially the case for FIR filters, which are not factored in the numerator and do not use Bij.* If denominator factors=1 then line p+2=1, line p+4 should equal "N" in the above equation and the pole coefficients are thus in fully expanded form. For an IIR coefficient file, if numerator factors or denominator factors is greater than one, then the Recursive Filtering function may or may not expand the corresponding factors prior to applying the filter (see the BIQUAD CASCADE field in section 4.9.1, Applying Filters with Recursive Filtering).**

Numeric Line Formats:

integer:	**mmmmm** where there are up to 5 m digits, preceded by a "–" only if the value is negative. There should be no leading blanks.
floating-point:	**L.mmmmmmmmmmmmmmESnnn** where there are up to 14 m digits, E is a literal and must be present as an "E", and Snnn is a sign (S *must* be either "+" or "–") and nnn is an exponent ≤ 307. For values less than zero a "-" (minus sign) must precede the L digit. "E" followed by exponent is not required if the exponent is zero. .mmmmmm... is not required if there the fractional part of the value is zero. There should be no leading blanks.

Comment Line Format:	Comment lines can be anything as long as they are terminated with a CR,LF (carriage-return, line-feed) sequence. Blank lines (CR,LF only) are completely acceptable.

4.8.3.2 Filter Specification File Format

Filter specification files are stored on disk as “filename.FFD” for FIR filters and “filename.IFD” for IIR filters, where “filename” is the name of your filter entered in the Hypersignal FIR or IIR Filter Construction data-entry menu. These files are ASCII, and divided into two sections. The first is basically a “notebook”, or documentation section that saves the menu parameters—this section is used by Hypersignal to initialize the data-entry menu when you first key in a filter name. The second section contains the actual frequency-amplitude pair values which describe the desired frequency response curve of the filter. For classical design methods (elliptic, Butterworth, Chebyshev, Kaiser Window, and Bessel) this section is ignored. An example file is shown below to illustrate the format (The comments to the right of each line are for illustration purposes only; *no comments should be entered into an actual file*. Also, although the lines of the file have been spaced apart for clarity in the description below, *lines should be single-spaced in the actual file.)* Finally, please remember that both frequency and amplitude values are stored in the file normally in the range of 0 to 1, inclusive, and are represented linearly—even though actual Hypersignal display can be linear or log on either axis. Values greater than 1 (for filters with gain > 1) can also be entered; see “Some Suggestions For Arbitrary Filter Editing”, in section 4.8.2.1 for instructions on editing gain values that are greater than unity. Thus, with this storage scheme, to enter an amplitude value of 0 dB, the linear value would be 1; a value of -40 dB would mean a linear value of 0.01 (calculated as 20∗log10[0.01]=-40). In the case of frequency values, 0 corresponds to dc, and 1 corresponds to 1/2 the sampling frequency that will be used.

E	design type: B=Butterworth BES=Bessel C1=Chebyshev C2=Chebyshev II DIF=differentiator E=elliptic HIL=Hilbert transform HM=Hamming Window KW=Kaiser Window PM=Parks-McClellan S=Steiglitz
50000	sampling frequency, in Hertz
15000	center frequency, in Hertz
10000	bandwidth, in Hertz
400	transition bandwidth 1, in Hertz
400	transition bandwidth 2, in Hertz
45	stopband attenuation, in dB
6	IIR filters: used only for Steiglitz filter order FIR filters: length of filter (number of taps)
2	passband ripple, in dB
77	total number of frequency-amplitude pair points that follow; *must be exact* (see notes below)
0.00000000000000E+00 5.62341325190349E-03	1st desired frequency point, 1st desired amplitude point
1.92000000000000E-02 5.62341325190349E-03	2nd desired frequency point, 2nd desired amplitude point
3.84000000000000E-02 5.62341325190349E-03	3rd pair
1.00000000000000E+00 5.62341325190349E-03	last pair (in this example, 77th pair)

1	one-byte parameter controlling display of desired and actual response onscreen (also controlled via 'O' key inside graphical filter design display): 0=only desired response should be shown 1=desired response and previous actual response should be "overlaid" (shown together) 2=only previous actual response should be shown
513	two-byte parameter controlling log/linear axes and grid display inside graphical filter design (also controlled via 'G' and 'L' keys inside graphical filter design display): high-byte values: 0=grid disabled 1=solid grid enabled 2=intersected grid enabled low-byte values: 0=log freq. axis, log vertical axis 1=linear freq. axis, log vertical 2=linear freq. axis, linear vertical 3=log freq. axis, linear vertical

NOTES: **Maximum number of pairs is 101. *Do not exceed 101!***

Amplitude values are *linear*, relative to a unity gain value of one (1). If you desire a certain dB value, then use the following equation to determine the value to enter into the file:

amplitude value = $10^{(dB/20)}$

where "dB" should be a negative number such as -12, -25, -65.5, etc. specifying the "dB down" from unity gain (zero dB) that the filter response should be at the associated frequency. "dB up" values can also be entered, which will translate into linear values greater than 1. See section 4.8.2.1, Using the Graphical Filter Editor, for instructions on editing gain values greater than 0 dB.

To make a frequency-amplitude pair display as "undefined", subtract 4 from the desired ampltitude value (the program "defines" an undefined point by adding 4 to the value).

Frequency-amplitude pairs should be ASCII floating-point, while all other parameters should be ASCII strings or ASCII decimal (integer) values.

4.9 Applying Filters In Hypersignal

Filters can be applied and tested in several ways in Hypersignal. FIR filters can be applied using the Convolution function, which convolves the filter's impulse response in non-real-time with an input waveform file, creating an output waveform file. Also, FIR filters can be "snapped-in" to the Digital Oscilloscope, Spectrum Analyzer, and Analog Conversion functions (real-time software only) and implemented in real-time on a supported DSP board. The Digital Oscilloscope and Spectrum Analyzer functions allow two filters to be applied at the same time.

IIR filters can also be applied using the Convolution function, but are more appropriately applied using the Recursive Filtering function. The impulse response for IIR filters in Hypersignal is created through estimation and must be truncated at some point, since by definition, an IIR filter has an "inifinite impulse response". Hypersignal creates very long IIR impulse responses in order to minimize the effects of truncation. Thus, using the Convolution function to convolve IIR filter impulse responses with input waveform files requires an unnecessary amount of processing time compared to the Recursive Filtering function. The Recursive Filtering function implements IIR filters using their "zero" (numerator) and "pole" (denominator, feedback) transfer function coefficients directly, thus fully utilizing the inherent speed advantage of IIR filters. Also, IIR filters can be "snapped-in" to the Digital Oscilloscope, Spectrum Analyzer, and Analog Conversion functions (real-time software only) and implemented in real-time on a supported DSP board. The Digital Oscilloscope and Spectrum Analyzer functions allow two filters to be applied at the same time. In this case, as with the non-real-time Recursive Filtering function, an appropriate IIR "cas-

cade structure" implemented in optimized DSP assembly language code is used that applies the transfer function coefficients directly, achieving maximum filtering speed.

Please note that there are several examples of applying filters in the Tutorial, including step-by-step menu set-ups and resulting waveform displays.

4.9.1 Applying Filters with Recursive Filtering

The Recursive Filtering function is used to apply IIR filters to input time domain waveforms. This function applies Z-domain transfer function coefficients directly, requiring an ".IIR" filter coefficient file, generated either by the Hypersignal IIR Filter Construction function or by an external program outside of Hypersignal. Filter coefficient files are stored in a very flexible, easy-to-read ASCII format that encourages user-modification and creation—see section 4.8.3.1 above, Filter Coefficient File Format, for full detail.

Upon completion of the Recursive Filtering function, the FILENAME2 field of the Waveform Display/Edit function will be initialized to the output time domain waveform, and the FILENAME1 field to the input waveform. This initialization allows "comparison viewing" between the unfiltered and filtered waveforms to be performed.

Recursive Filtering Menu Fields

The paragraphs below explain the Recursive Filtering data-entry fields:

INPUT DATA FILE x[n]:

This field specifies the filename of the input time domain waveform. If not found, a directory of ".TIM" files will appear in the System Status area; use the 'PgUp' and 'PgDn' keys to search for a waveform (see section 3.4.2, Pop-Up Directory Display).

COEFFICIENT (FILTER) FILE:

This field specifies the filename of the input filter coefficient file. Normally this will be a file produced by the IIR Filter Construction function, but is stored in a readable ASCII text format, and thus may also be user-generated. See section 4.8.3.1 for a file format description. This field is initialized to the previous IIR Filter Construction function output.

If this field is not found, consecutive use of the 'PgUp' key will display pop-up directory information for ".IIR" filter coefficient files in the System Status area.

OUTPUT DATA FILE y[n]:

This field specifies the filename of an output time domain waveform file that will contain the resulting filtered waveform.

BIQUAD CASCADE:

A 'Y' entry enables a "biquad" IIR filter cascade structure for filter implementation, which requires both the numerator and denominator of the filter transfer function to be presented in the filter coefficient file in the form of quadratic (2nd-order or less) factors. This is the form used for IIR Filter Construction output by the Hypersignal software, and provides the greatest stability for high order filters and/or filters with near-oscillatory poles or zeros (poles or zeros very near the imaginary axis in the S-plane, or very near the unit circle in the Z-plane).

An 'N' entry disables the biquad cascade structure, and instead causes both numerator and denominator to be fully expanded, regardless of the order of each individual section(s) as stored in the filter coefficient file. This allows for maximum flexibility in the case of a user-entered coefficient file (see file format above), where filter numerators and denominators can be entered in forms ranging from a single polynomial to single-order sections.

NOTE: Theoretically, different methods of filter implementation should make no difference in output if they are truly equivalent. However, in practice even 6-byte or 8-byte real precision may not be enough to support multiplying out all filter coefficients beforehand (expanding the factors) of very high-order or 'sensitive' filters; thus, biquad cascade structure is recommended whenever possible.

4.9.2 Applying Filters with Convolution

The Convolution/Correlation function allows two time domain waveforms to be convolved or correlated with one another, producing a third time domain waveform. Although any two time domain waveforms may be specified, Hypersignal has been designed to make this selection particularly useful when convolving a waveform with a filter designed using the FIR Filter Construction function. It is also possible to use the output impulse response from IIR Filter Construction as the filter input, although the Recursive Filtering function (see section 4.8.2, Applying Filters with Recursive Filtering) is intended specifically for this purpose.

Upon completion of the Convolution/Correlation function, the FILENAME2 field of the Waveform Display/Edit function will be initialized to the output time domain waveform, and the FILENAME1 field to the input waveform. This initialization allows "comparison viewing" between the unfiltered and filtered waveforms to be performed.

Convolution/Correlation Menu Fields

The following paragraphs explain the data-entry menu fields in the Convolution/Correlation function:

INPUT DATA FILE x[n]:

This field should specify the filename of the input time domain waveform to be convolved or correlated with the h[n] (filter) waveform. If not found, a directory of ".TIM" files will appear in the System Status area; use the 'PgUp' and 'PgDn' keys to search for a waveform (see section 3.4.2, Pop-Up Directory Display).

IMPULSE (FILTER) FILE h[n]:

This data field should specify the name of the file containing the input time or filter waveform to be convolved or correlated with the file x[n] above. Normally this will be a file produced by the FIR Filter Construction function, but may be any time waveform file. If the file entered contains more than 16384 samples, only the first 16384 samples will be used in the convolution (only the first 8192 sample pairs will be used for a stereo convolution).

This field is initialized to the output filter filename from the previous FIR Filter Construction.

If this field is not found, consecutive use of the 'PgUp' key will display pop-up directory information for ".TIM" time domain waveform files in the System Status area.

OUTPUT DATA FILE y[n]:

This field should specify the filename of a time domain waveform file that will contain the resulting convolution/correlation output.

CONVOLVE/CORRELATE:

Possible entries for this field are:

'-'	specifies convolution operation
'+'	specifies correlation operation
'A'	specifies autocorrelation operation

ARBITRARY FILTER SCALING:

This field is only active when the IMPULSE (FILTER) FILE field above specifies a waveform that has its TIME DOMAIN SOURCE attribute (see chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT) set to arbitrary time domain data. Usually, the Convolution/Correlation function will be used with IMPULSE FILE waveforms that have their TIME DOMAIN SOURCE attribute set instead to transfer function, or "H[n]", data. The Filter Construction menu selections and writing transfer function display traces from the Spectrum Analyzer are two examples of built-in Hypersignal functions that automatically create waveform files with TIME DOMAIN SOURCE set to "H[n]". Arbitrary scaling becomes necessary when nothing is known about the frequency response and gain characteristics of the filter waveform file (see "Notes About the Convolution/Correlation Process" below). Possible entries for this field are:

'RMS'	specifies a scaling factor for arbitrary filters based on an RMS calculation of time domain filter coefficients (see Notes below)
'EST'	specifies a scaling factor for arbitrary filters based on frequency content

The default, and recommended entry for this field is 'RMS'.

Notes About the Convolution/Correlation Process

1) Frame Counter

The "FRAME xx OUT OF XX" counter displayed is intended for counting (user feedback) purposes only, and has no bearing on actual processing.

2) Framesize

The FRAMESIZE value stored in the impulse (filter) waveform file header should be equal to the length of filter waveform. If not, you will be prompted as to whether to use the FRAMESIZE value as the length of the filter or the length of the waveform. In either case, the length of the filter cannot exceed the maximum values given above.

3) Inherent Delay

Convolution can be mathmatically expressed as:

$$y[n] = \sum_{k=0}^{K-1} h[k]\, x[n-k]$$

and correlation as:

$$y[n] = \sum_{k=0}^{K-1} h[k]\, x[n+k]$$

where K is the finite length of the filter sequence h[k]. In the case of convolution, note that until "n" (the current time series sample index) exceeds K-1, the equation will reference *negative-time samples* for the x[n] sequence. For correlation, values of "n" exceeding the length K of the input waveform x[n] will also reference undefined values. Because Hypersignal always expects to deal with real-world signals that are not defined before time zero and have finite length, it evaluates x[n] for $n<0$ or $n>K-1$ as zero. Thus, for convolution there is a built-in delay in the y[n] output waveform of N samples before the filter is being applied consistently to the x[n] input waveform. *This is normal—do not edit-out these delays* and then expect to see correct theoretical results; for example, a Hilbert transform FIR filter, which is designed to inject a 90-degree phase lag into an output y[n] sequence from an input x[n] sequence, will operate correctly *after* this delay, even though the y[n] waveform may look a little rough at its beginning when viewed in the Waveform Display/Edit function. See the Tutorial for an excellent example of Hilbert transform design and verification.

4) Autocorrelation

Autocorrelation can be mathmatically expressed as:

$$y[n] = \sum_{k=0}^{K-1} x[k]\, x[n+k]$$

where K is the finite length of the *frame of the input sequence x[k] that should be autocorrelated with the remainder of x[n]*. When the autocorrelation option has been specified in Hypersignal, the length K starts at the FRAMESIZE value of the impulse (filter) waveform, and then decreases by one sample after each value of y[n] is calculated. When K reaches zero (when FRAMESIZE output values have been calculated), it is reset to FRAMESIZE. This repeats after every FRAMESIZE output values are calculated.

5) Convolution Scaling

When using the output of FIR Filter Construction function or the Spectrum Analyzer transfer function storage option, the Convolution function makes use of the MAX AMPLITUDE and AMPLITUDE EXPONENT parameters stored in the filter waveform file header to apply FIR filters so that unity gain is associated with 0 dB in the frequency response of the filter. A filter

waveform scaling factor is created by dividing the MAX AMPLITUDE of the filter waveform by 32767 and then multiplying by $2^{\text{AMPLITUDE EXPONENT}}$. This is done as the inverse to output scaling performed by the FIR Filter Construction and Spectrum Analyzer functions when writing out time domain filter data (taps, or coefficients).

However, when an arbitrary time domain waveform is used as the h[n] waveform, there is no pre-existing knowledge about the waveform that ensures unity gain in whatever "passband", or "0 dB", area that might exist in the frequency response of the arbitrary waveform. Therefore, the Convolution/Correlation can make two types of estimation of the appropriate scaling factor to adjust the coefficients by: one based on an RMS calculation using the time domain coefficients, and another based on a frequency content estimate.

The RMS calculation is an estimation of the power contained in h[n] made using Parseval's theroem. All coefficients are squared, summed, averaged, and then the squareroot is taken to return the final scaling factor. Theoretically, the resulting output y[n] waveform is then scaled so that it will contain no more total power than the input waveform x[n]. However, the y[n] output waveform may exhibit certain frequencies where gain exceeds unity, indicating that some frequencies have gained energy at the expense of others.

The frequency content estimation of the h[n] waveform method for determination of the appropriate arbitrary filter scaling factor is still under evaluation and is not currently recommended for use.

4.9.3 Using "Snap-In" Real-Time Filtering

(Real-Time Software Only)

In the Analog Conversion, Digital Oscilloscope, and Spectrum Analyzer functions, you can "snap-in" an FIR or IIR filter via the data-entry menu. When you do this, the filter is applied in real-time to the input (or output) data stream by the DSP device on the current DSP/Acquisition hardware. For fixed-point DSP devices you can control coefficient quantization via the data-entry menu also.

By specifying a filter coefficient filename, you direct the software to read in IIR coefficients or FIR "taps" and download them to the DSP board memory. The filter coefficient file is normally created by the Filter Construction functions in Hypersignal, but can also be user-created, outside of Hypersignal (see section 4.8.3 Filter Coefficient File Format).

NOTE: If neither ".IIR" nor ".FIR" filter coefficient files cannot be found, the software attempts to locate a filename with a ".TIM" extension. If such a waveform file is found, then the waveform values are "converted" to FIR filter coefficients using the Parseval scaling described in section 4.9.2, Applying Filters with Convolution. Arbitrary waveforms can thus be convolved with incoming signals in real-time, using this method.

For the Digital Oscilloscope and Spectrum Analyzer functions, for analog input only, enter the filter coefficient filename after the 'An' input channel entry in the TRACE1 or TRACE2 fields (use a comma as a separator). For example, you might enter TRACE1 and TRACE2 fields of:

TRACE1: a0,filt1
TRACE2: a1,filt2

This example will apply the "filt1" filter to channel 0, and the "filt2" filter to channel 1. As another example, you might enter in the Spectrum Analyzer function:

TRACE1: filt1
TRACE2: a0,filt1

This example will show the waveform file response for the "filt1" filter on trace 1 and apply the "filt1" filter to channel 0, showing it on trace 2. If you then input a repeating swept sine wave to channel 0, you would be able to compare theoretical and actual filter responses using your DSP board. As a another example of instrumentation entry, you might enter:

TRACE1: a0
TRACE2: a0,filt1

This example will simply compare unfiltered and filtered versions of channel 0 input on trace 1 and trace 2, respectively.

For fixed-point DSPs only, a quantization value can also be entered after the filter filenames. Consider the following instrumentation example:

TRACE1: a0,filt1,12
TRACE2: a1,filt2,12

Unless explicitly entered, the choice of number of bits to quantize filter coefficients to is handled automatically by the software, which selects the highest "Q-notation" possible, depending on the filter coefficients. Note that *quantization values must always be the same for two simultaneous filters.* If you only enter one quantization value when using two filters, then the other filter will use this value also, even though nothing has been entered for it.

For the Analog Conversion function, enter the filter coefficient filename after the last entry in the CHANNEL LIST field (use a comma as a separator). A quantization value can also be specified (use another comma). For example, you might enter a CHANNEL LIST field of:

CHANNEL LIST: 0,1,elliptic

This example will apply the "elliptic" filter to channels 0 and 1. Unlike the instrumentation display examples above, the form of entry used in the Analog Conversion function does not allow different filters to be applied to different channels independently. Only one filter can be specified—and it is either applied to one channel or two (or more if the DSP source code has been modified—see the Hardware Reference Guide). No filter is applied to a channel if it does not appear in the CHANNEL LIST field.

As another example, you might enter a CHANNEL LIST field of:

CHANNEL LIST: 0,butter,14

This example will apply the "butter" filter to channel 0, and also force the coefficient quantization to be 14 bits.

For fixed-point DSPs, unless explicitly entered, the choice of number of bits to quantize filter coefficients to is handled automatically by the software, which selects the highest "Q-notation" possible, depending on the filter coefficients.

For IIR filters, Hypersignal initializes the DSP code to one of two different types of cascade structures: "structure 2", appropriate to all-pole filters (Butterworth, Chebyshev I, Bessel), or "structure 3", appropriate to pole-zero filters (elliptic, Chebyshev II, Steiglitz arbitrary). See section 4.9.5, Generating DSP Code for Filters, for details about IIR cascade structures.

NOTE: If you would like to use a different IIR cascade structure than the Hypersignal default, you can change the structure field in the IIR filter coefficient file directly using a text-editor. See section 4.8.3.1, Filter Coefficient File Format, for more detail.

If you require other DSP assembly language changes such as rounding, or think you can eke out more speed from the filter routines (we would be very impressed if you did), then you can always make the necessary changes directly in the appropriate "DSP Source Code Interface" filter routines (see the Hardware Reference Guide).

NOTE: Please remember when you use real-time filters that maximum sampling rates for the function involved will drop, because the DSP is now working harder between sample interrupts, using more processing time. The amount of decrease will depend on a number of factors, especially the length of the filter, speed of the DSP chip, and efficiency of the chip in implementing "real-world" digital filter structures (not the trade-journal benchmark kind, which never work with a filter anyone would actually use). If you exceed the capability of your DSP device, you may lock up the function altogether—try decreasing your sampling rate if you experience no display or no conversion to/from disk. And yes, as shown in the above examples, you can apply independent filters to different channels in the Digital Oscilloscope and Spectrum Analyzer functions. However, when you do this, keep in mind that the DSP is now working twice as hard (which will further decrease the maximum rate at which the function can stay in real-time). See chapter 6, BRIEF DISCUSSION OF SAMPLED DATA SYSTEMS, for background information on real-time processing.

Depending on the type of DSP chip in use, the maximum length of FIR filters (or maximum number of sections for IIR filters) could be limited by the amount of DSP on-chip RAM available. Of course, with proper modification of the DSP source code, you can move filter coefficient and delay element addresses to other, larger areas of memory. As far as this issue is concerned, the high-level software acts relatively dumb: it only loads filter coefficients where the DSP source code indicates, and it loads as many coefficients as are found in the Hypersignal filter coefficient file (which means you can easily crash the DSP program if the filter is too big for the current memory area).

Without DSP source code modification, maximum filter lengths are as follows for different DSPs:

#, type	TMS320C30	DSP56001	TMS320C25	DSP32C
1 FIR filter	taps=512	taps=431	taps=64	taps=2048
2 FIR filters	taps=256	taps=215	taps=32	taps=1024
1 IIR filter	order=40*	order=40*	order=24	order=40*
2 IIR filters	order=40*	order=40*	order=12	order=40*

* 40 is Hypersignal program limit for IIR filter design, as well as other functions that use either Hypersignal-generated or user-defined filter coefficient files as input.

When using filters in the Analog Conversion function with "monitor" mode enabled (digital loopback of input to output) the output will be filtered. Thus, should you start an Acquisition process, but never answer the "BEGIN ACQUISITION (Y/N)?" prompt, you will turn your computer into an oversized real-time filter.

The filter routines in the DSP source code are basically derived from the Hypersignal Code Generate/Quantize function with the following alterations: variable quantization if the DSP chip is fixed-point (i.e. quantization can be dynamically modified in the filter routines), no rounding, and the addition of complete context-save. For some fixed-point DSPs, changing quantization may involve somewhat unorthodox methods, such as dynamically modifying instruction memory op-codes, in order to maintain maximum filter execution speed.

4.9.4 Using Pole-Zero Analysis of IIR Filters

Pole-Zero analysis of IIR filters is accomplished in Hypersignal using the Pole-Zero Plot frequency domain display menu selection. This selection allows the poles and zeros listed in an IIR filter coefficient file to be viewed graphically in the S-plane or Z-plane.

The input filter coefficient file (".IIR" filename extension) is normally created by the Hypersignal IIR Filter Construction function, but can also be easily generated outside of Hypersignal. Filter coefficient files are stored in a readable, well-documented ASCII text format; see section 4.8.3, Specifying Filters Outside of Hypersignal, for more information.

NOTE: If you do wish to define an IIR filter coefficient file outside of Hypersignal for Pole-Zero Plot analysis, there is one restriction: because of the complicated math required to perform an inverse bilinear transformation, the maximum section (factor) order in the coefficient file must be 2 (2nd-order cascade sections). Both single and second-order sections are allowed; if the third coefficient of a second-order section is zero, then it is defined as a single-order section.

In the Z-plane, 2nd-order factor numerator and denominator pairs ("biquads") are solved directly for their poles and zeros and plotted relative to the unit circle.

In the S-plane, an inverse bilinear transformation is first required to calculate poles and zeros. This transformation is based on the assumption of normalized filters in the S-plane. Both lowpass and highpass filter cutoff frequencies are normalized to $\pi/2$, while bandpass and bandstop filter cutoff frequencies are normalized to $\pi/4$ and $3\pi/4$ for lower and upper passband/stopband cutoff frequencies, respectively. Although any transformation may be chosen, correct (and meaningful), display of S-plane poles and zeros depends upon choosing the appropriate transformation (i.e. lowpass transformation for a lowpass filter, etc.). The software automatically initializes this data-entry field from the IIR filter coefficient file header.

Pole-Zero Plot Menu Fields

The following data-entry fields may be modified and entered in the Pole-Zero Plot menu:

FILENAME:

This field should specify the filename of an input IIR filter coefficient file for pole-zero analysis.

If the FILENAME specified is not found, then a directory of ".IIR" files will be displayed in the System Status area. Paging through the directory is accomplished by using the 'PgDn' key to scroll down, and the 'PgUp' key to return to the first page of the directory. The search strategy for pop-up directory display will change with consecutive use of the 'PgUp' key—see section 3.4.2, Pop-Up Directory Display, for details.

PLANE:

This field should specify whether poles and zeros are to be viewed in the S-plane, Z-plane, or both simultaneously. Possible entries are:

'S'	selects S-plane display
'Z'	selects Z-plane display
'B'	displays both S-plane and Z-plane simultaneously (software default)

INVERSE TRANSFORMATION:

This field should specify which inverse bilinear transformation to use in calculating S-plane poles and zeros. Although normally initialized automatically by the software from the input IIR filter coefficient file header, attention should be paid to ensure use of the correct transformation. Possible entries are:

'LP'	lowpass inverse transformation
'HP'	highpass inverse transformation
'BP'	bandpass inverse transformation
'BS'	bandstop inverse transformation

For Steiglitz arbitrary IIR filters, a 'BP' value is placed in the IIR filter coefficient file header by the design process as a good approximation to the mixed pole-zero nature of Steiglitz filters.

Scaling and Plotting Notes

Poles and zeros are scaled to the unit circle when represented in the Z-plane. Unstable poles/zeros (i.e. outside unit circle), are drawn in inverse video.

Poles and zeros are scaled to a circle of radius π when shown in the S-plane. The maximum S-plane pole or zero magnitude is then displayed as "Max Wc Radius", and indicates the filter's *digital* cutoff frequency (not its analog, which has since been normalized through filter design such that π represents the original "Nyquist rate", or analog sampling frequency divided by two). In this way, filter cutoff frequency can be represented as the distance of poles and zeros from the edge of the circle. Autoscaling is in effect if for some reason, such as incorrect inverse bilinear transformation, a pole or zero has magnitude greater than π, resulting in scaling to a circle of radius equal to the largest pole or zero magnitude. S-plane poles and zeros at infinity are not plotted. Unstable S-plane poles and zeros (i.e. right half of S-plane) are drawn in inverse video.

4.9.5 Generating DSP Code for Filters

Using the Code Generate/Quantize menu selection, Hypersignal can generate optimized assembly language or 'C' language source code for several different DSP devices based on filter coefficient file input. This source code is ready-to-assemble or compile, and well-commented. Source code can be generated for both IIR and FIR filters; in addition, one of three different IIR cascade structures is possible. An option to generate a simulated quantized impulse response suitable for further analysis is also included.

DSP chips for which source code can be generated include:

Manufacturer	DSP Device
Analog Devices	ADSP-210x (16-bit fixed-point), ADSP-21020 (32-bit floating-point)
AT&T	WE DSP32 (32-bit floating-point), WE DSP32C (32-bit floating-point)
Motorola	DSP56001 (24-bit fixed-point), DSP96002 (32-bit floating-point)

NEC	77220 (24-bit fixed-point)
Texas Instruments	TMS320C10 (16-bit fixed-point), TMS32020 (16-bit fixed-point), TMS320C25 (16-bit fixed-point), TMS320C30 (32-bit floating-point)

Filter coefficient files are normally created by either the Hypersignal IIR or FIR Filter Construction functions, but can also be easily generated outside of Hypersignal (see section 4.8.3, Specifying Filters Outside of Hypersignal).

Options which may be specified when generating assembly language source code include IIR filter cascade structure, coefficient base addresses, memory mode, rounding, quantization, and generation of a "simulated quantized impulse response" for subsequent analysis of the quantization effect on the filter.

Source code is *generated for the DSP chip entered in the* DEVELOPMENT DSP *field* in the System Configuration menu selection. Entries for this field include:

'AD21'	Analog Devices ADSP-210x
'AD210'	Analog Devices ADSP-2102x
'WE32'	AT&T WE DSP32
'WE32C'	AT&T WE DSP32C
'N20'	NEC 77220
'M56'	Motorola DSP5600x
'M96'	Motorola DSP9600x
'TI10'	Texas Instruments TMS32010, 320C10, 32011, 320E17, etc.
'TI20'	Texas Instruments TMS32020
'TI25'	Texas Instruments TMS320C25
'TI30'	Texas Instruments TMS320C30

The generated code includes "commented-out" main program sections that create an impulse response by calling the filter routines, and may be used for filter testing by assembling and linking using the manufacturer's development utilties. All generated assembly language source code is designed to be assembled by the *manufacturer's development tools;* others must be truly compatible. "C" language source must be compiled by a Kernighan & Ritchie standard C compiler.

Code Generate/Quantize Menu Fields

The following data-entry fields can be modified and entered in the Code Generate/Quantize menu:

FILENAME:

This field should specify the filename of an input IIR or FIR filter coefficient file, and also the filename of the output source code file. Output source code files have ".S" filename extensions for AT&T devices, ".DSP" for Analog Devices devices, ".ASM" for Motorola and Texas Instruments devices, ".SRC" for NEC devices, and ".C" for C language source code.

If the FILENAME specified is not found, then a directory of ".?IR" files (both ".FIR" and ".IIR" files) will be displayed in the System Status area. Paging through the directory is accomplished by using the 'PgDn' key to scroll down, and the 'PgUp' key to return to the first page of the directory. The search strategy for pop-up directory display will change with consecutive use of the 'PgUp' key—see section 3.4.2, Pop-Up Directory Display, for details.

NOTE: The IIR filter construction function will not allow construction if FIR files with the same name already exist, and vice versa, thus minimizing chances of filename confusion. If external, manual file-manipulation should cause both FIR and IIR filenames to exist, a prompt asking you to choose will appear.

IIR STRUCTURE:

This field should specify the structure for implementing cascade 2nd-order IIR filter sections, and is ignored for FIR filters. The software will initialize this field appropriately for you based on input from the filter coefficient file header. Possible entries are:

'S1'	Structure 1. Specifies a canonic direct-form structure that requires the theoretical minimum amount of multiplications for implementing IIR cascade 2nd-order "biquad" sections. Although the fastest structure, S1 is also the most unstable for fixed-point devices, because quantization error is not distributed evenly throughout the filter. When using a floating-point device, S1 will work adequately in most cases. *This structure is definitely not recommended for any type of filter when using a fixed-point device.*
'S2'	Structure 2. Specifies a "modified direct-form" structure, basically derived from structure 1 above. In this structure, the "constant multiplier" for the filter is distributed evenly throughout the numerator coefficients of each biquad section. This structure is best-suited to "all-pole" filters (filters with no zeros in their S-plane transfer function), which include Bessel, Butterworth, and Chebyshev I.
'S3'	Structure 3. Specifies a "balanced pole-zero" structure that attempts to locally match pole-zero pairs in an attempt to cancel some quantization error. This structure is best-suited to "pole-zero" filters (filter with both poles and zeros in their S-plane transfer function), which include Chebyshev II, elliptic, and arbitrary Steiglitz designs.

NOTE: **For very high-order filters or "sensitive" filters with sharp cutoff frequencies or poles very near the imaginary axis in the S-plane, even 32-bit floating-point precision may be inadequate. In such cases, you might follow the fixed-point guidelines mentioned above and use structures 2 and 3. This is even more important for 24-bit fixed-point devices, which perform well in most cases, but still allow a substantial possibility for instability in filters that rely on pole-zero balancing to achieve sharper cutoffs, namely elliptic and Chebyshev II-type filters.**

LANGUAGE:

This field should specify the type of source code output desired. Entries can be:

'ASM'	causes output source code to be in native assembly language, depending on current value of DEVELOPMENT DSP field in the System Configuration menu selection
'C'	causes output source code to be in general 'C' language form not specific to any DSP chip or compiler

QUANTIZATION:

This field should specify the number of bits to which the filter coefficients and delay elements should be quantized for fixed-point processors, *and is ignored for floating-point processors.* Quantization, sometimes referred to as "Q-Scaling" or "Q-Notation", causes Hypersignal filter file floating-point coefficients to be quantized to fixed-point values ranging from 9 to 32 bits, depending on the "Q" number entered in this field. Entry is in the form:

'Qnn'

where "nn" is a number ranging from 8 to 31 (9-bit values, including sign bit, to 32-bit values). For fixed-point DSP devices, the maximum value of "nn" is implied by the current entry in the DEVELOPMENT DSP field in the System Configuration menu. For example, if a 16-bit fixed-point device is specified in the DEVELOPMENT DSP field, then 'Q15' is the maximum entry allowed. If no DSP device (a 'none' entry) is selected in the DEVELOPMENT DSP field, then the maximum quantization entry is 'Q31'. The 'none' entry in the DEVELOPMENT DSP field can be useful for performing chip-independent experimentation with quantization values. Software default entries are 'none' for the DEVELOPMENT DSP field and 'Q14' for the QUANTIZATION field.

If you receive the prompt "QUANTIZATION TOO HIGH..." after initiating code generation then at least one numerator or denominator coefficient exceeded "1" (exceeded the number of maximum bits) for the quantization chosen. If the coefficient magnitude is exactly or very close to 1.0, then responding to the prompt with a 'C' (CLIP) will cause minimal error, as the coefficient(s) will be truncated to + or -0.9999999... using the chosen quantization. Otherwise, you should respond to the prompt with a 'Q' (QUANTIZE ADJUST), which will cause the software to automatically adjust the quantization downward.

NOTE: **For some fixed-point processors, *some values of quantization are very expensive* in terms of code execution-time because they deviate from the "natural word-length" used by the device. The penalty paid for non-natural quantization values ranges from none to substantial, depending on the DSP chip (see the discussion in "Real-Time Processing Speed vs. Quantization", below). Unfortunately, many IIR filters designs produce coefficients that are too large for natural word-length quantization—something the DSP chip manufacturers conveniently avoid in their rather simple benchmarks. For IIR filters, you will have to experiment with different quantizations and structures, and compare generated assembly language listings, to see how your particular DSP device is affected. At least it's better to be able to do this ahead-of-time, without actually running any code yet.**

COEFFICIENT BASE ADDRESS / MODE:

Depending on the current DSP device, this field should specify the base address for filter coefficients or the memory mode that should be used. Below is a summary of how this field is interpreted for different DSP devices:

TMS32010, TMS32020	filter coefficient base address in external data memory
TMS320C25	filter coefficient base address in on-chip data memory
DSP5600x	filter coefficient address in on-chip X: data memory
AT&T WE DSP32, DSP32C	desired memory mode (for DSP32, filter coefficients are stored in external program memory; for DSP32C, coefficients are stored in on-chip data memory)
ADSP-210x	field is ignored
NEC77220	field is ignored

Memory mode values should be entered in this field as a single decimal number. Coefficient base addresses can be entered in this field as a single hexadecimal number, or in "PAGE:OFFSET" form. Some base address entry examples:

'0'
'0300'
'6:0'
'1FE:40'

When the PAGE:OFFSET format is used, the size of a page is the natural value used by the currently selected development DSP (e.g. 128 words for TI 320xx family processors). In cases where memory paging is not supported by the currently selected DSP, the PAGE:OFFSET format should not be used.

Below are more detailed COEFFICIENT BASE ADDRESS field entry instructions for specific DSPs:

TMS32010, TMS32020, TMS320C25 FIR	All coefficients (both IIR and FIR) must be in *on-chip data memory* page 0 (zero). IIR coefficients can reside on pages 4-510 (pages 0-3 are reserved, and page 511 has been saved for user variables). *Remember that pages 8-511 are in external data memory,* which may or may not be in your target system. FIR coefficients are stored in *program memory* (on-chip block B0, after invoking the CNFP instruction).
TMS320C25 IIR	Coefficients can reside in on-chip data memory only. This is because the filter structures generated by Hypersignal utilize the "LTD" instruction, which performs data moves in parallel with arithmetic operations, but only in fast on-chip RAM. If you require filter code that uses external data memory for coefficients and delay elements, generate code for the TMS32020 and then combine/replace instructions indicated in the comments with their faster TMS320C25 counterparts.
TMS320C30	BASE COEFFICIENT ADDRESS / MODE is ignored. All filter coefficients and delay storage elements are stored in on-chip RAM.
DSP5600x	Filter coefficients are stored in on-chip X: data memory at this address *only for a non-zero entry*. For zero entry, use of X: memory is made relocatable. Filter delay storage elements are stored in on-chip Y: data memory, which is left relocatable always. An absolute starting address for Y: memory can be added manually to the output source code if necessary.
DSP32	Memory MODE must between 0 and 3, inclusive. Default output source code locates delay elements in on-chip "lo_ram", thus requiring MODE 0,1, or 3. Locating *both* filter coefficients and delay elements in on-chip "lo_ram" or "hi_ram" in the final application code will minimize execution time.

DSP32C	Memory MODE must be between 0 and 7, inclusive. Has no effect on where coefficients and delay elements are stored (on-chip data memory), but does affect some instructions in the generated code. Use modes 4-7 to generate code that takes advantage of new DSP32C instructions.
ADSP-210x	BASE COEFFICIENT ADDRESS / MODE field is ignored. For the ADSP-2100, all filter coefficient (data memory) and delay storage elements (program memory) base addresses are defined by the user in a separate "system builder" .ACH (architecture) file. For the ADSP-210x filter coefficients are stored in onchip data memory at 3800H, and delay elements in program memory at 3C00H.
NEC77220	BASE COEFFICIENT ADDRESS / MODE field is ignored. All filter coefficients and delay storage elements are stored in on-chip RAM0 and RAM1 starting at location 0.

ASSEMBLY CODE ROUNDING:

This field specifies whether to enable ('Y' entry) or disable ('N' entry) fixed-point filter assembly language source code rounding, and is ignored for C language source code. For fixed-point source code, rounding can sometimes improve filter response accuracy and noise floor, but usually with an increased filter execution time penalty. Rounding basically amounts to adding 0.5 in the proper quantization prior to moving values from double-precision accumulators to single-precision memory locations.

A comparson of simulated quantized impulse responses (see below) is suggested to see if rounding will make any difference in your filter implementation.

SIMULATE QUANTIZED h[n]:

For fixed-point processors only, this field specifies whether to enable ('Y' entry) or disable ('N' entry) generation of a simulated quantized impulse response. The impulse response is generated by Hypersignal exactly as if it were being created on the current DSP device, using the current QUANTIZATION field setting, current IIR cascade structure (if applicable), and current ROUNDING field setting.

Entry is in the form:

'Y,INP' or 'N,INP'

where "INP" is floating-point value ranging from 0 to 1 that specifies the "input level" to use as an impulse. This value is converted by the software to the correct integer value using the current QUANTIZATION field setting, above.

The simulated quantized impulse response is output as a time waveform with "_Q" appended to FILENAME field entry, above (the last two characters of FILENAME are overwritten if FILENAME is longer than 6 characters). The quantized impulse response can be easily compared to the original impulse response and frequency response using Waveform Display/Edit, FFT Generation, and Magnitude Display functions. See section 4.9.5.1, Simulating the Effects of Quantization, below for program examples showing exactly how to do this.

NOTE: For IIR filters, you may need to change the input value depending on the QUANTIZATION field value and the nature of your filter. A typical value of 0.25 (which is the software default) allows for some signal growth during impulse response simulation. For very "sensitive" or marginally stable IIR filters (with poles/zeros near the imaginary axis in the S-plane), low levels of maximum input and/or lower quantization levels may be required to avoid overflow or instability.

Real-Time Processing Speed vs. Filter Order/Length

When expecting to run assembled filter code in real-time, remember that IIR filter order and FIR filter length affect processing speed.

For IIR filters, (N+1)/2 2nd-order stages are required to implement a given filter, where N is the filter order (note that if N is odd then one stage will reduce to first-order, even though the number of stages wouldn't change if N was increased by one).

For FIR filters, the concept of "stages" doesn't really apply; instead, X number of instructions are needed per filter coefficient (ipc), or X∗filter length total, plus some instructions necessary one time only, regardless of filter length.

Real-Time Processing Speed vs. Quantization

Default Hypersignal-generated filter code is always optimized for speed, but for Texas Instruments TMS320xx, Motorola DSP5600x, and Analog Devices ADSP-210x fixed-point code, some choices of quantization produce markedly slower code.

For the Texas Instruments TMS32010 Q15,Q14, and Q11 are fastest, with Q13-Q12 and Q10-Q8 taking almost twice as long. For the TMS32020, Q15, the natural choice for a 16-bit machine, is the fastest, with Q14 and Q11 only slightly slower, and Q12-Q10 slower still, and Q9 and Q8 progressively slower. For the TMS320C25, Q15-Q9 can all be implemented without any extra instructions. These quantization-speed differences on TMS320xx processors are due mainly to the inherent nature of the SACH instruction in the various processors, with the TMS320C25 coming closest to complete independence of quantization.

For Motorola DSP5600x and Analog Devices ADSP-210x processors, quantization values other than Q23 (Motorola) or Q15 (Analog Devices) forces much slower processing speed, because extra instructions that align intermediate results and cannot be "pipelined" to execute in parallel with other instructions must be inserted. Also, almost unbelievably, the DSP5600x can only shift one bit per instruction cycle, forcing the use of a hardware "DO loop" instruction to achieve multiple shifts, further slowing execution time. Unfortunately, many filters have coefficients that cannot be "Qmax-scaled" on these processors, and still others will not allow nominal input levels to be processed at maximum quantization without some signal overflow in intermediate results.

Assembly Language Source Code Organization

All generated code is organized in looped form (number of iterations equal to number of stages in IIR filter or number of taps in FIR filter), except for TI TMS32010 and TMS32020 IIR filters, which are organized as repeated blocks of inline code. In the case of all looped filters, if you require that last extra bit of speed (especially if the loop instruction itself is multi-cycle or otherwise high overhead), then of course the filter code may be replicated without the loop instruction for each stage. If procedure or variable names/locations conflict with pre-existing code, then it becomes a user responsibility to alter names and locations in order to ensure that all code meshes together and can be assembled error-free. Hypersignal does use variable location equate statements and indirect addressing wherever possible to aid in producing more relocatable code (quite helpful in the event of memory usage conflict with pre-existing code), and also to aid in readability and code maintenance.

NOTE: **Set-up instructions *prior to the main loop* inside filter routines can be moved from the filter routine, which must be invoked for every input sample, to a one-time-executable program location *if allowed by resident application code.* The philosophy behind these set-up instructions, which typically perform such functions as setting addressing mode registers and arithmetic defaults (sign extension, overflow/saturation, post-multiplication shifts, etc.), is "better to let the user conciously place them in a one-time-only, out of sight location, rather than assume they can be put there automatically" in the generated output source code. Also, filter routines do not perform a complete context-save of all processor status and registers. For source code examples of filtering with complete context-save, please consult the appropriate "DSP Source Code Interface"—see section 4.7, Using DSP/Acquisition Hardware with Hypersignal, and the Hardware Reference Guide for more information about DSP Source Code Interfaces for specific DSP chips.**

Testing Filters with Manufacturer's Software Simulator

If you wish to test your filter with the manufacturer's software simulator or hardware emulator, create an impulse such that the first input sample to the filter is non-zero (typically 0.1-0.5 in the appropriate quantization for fixed-point processors), and all other input is zero for the remainder of the length of the truncated infinite impulse response for IIR filters or exact finite filter length for FIR filters. In both cases, this length value is defined in the generated source code as a constant ("EQU", "const", etc. statement defining a name like "LEN", "filter_length" or similar).

All Hypersignal-generated code includes "commented-out" code that does exactly these steps and may be used for impulse response testing. These "main program" examples test filters by storing unit-impulse response output in memory, which may be written to disk using simulator or emulator utilities, and then acquired by the File Acquisition function for display and analysis.

4.9.5.1 Simulating the Effects of Quantization

Below is a series of program examples intended to illustrate how to simulate the effects of quantization on IIR and FIR filter impulse and frequency responses. Investigation of quantization effects may be needed prior to implementation on a specific DSP device, or for chip-independent algorithm development purposes.

The example sequence below starts with an IIR Filter Construction design, then displays this frequency response, then generates a 14-bit coefficient quantized impulse response, then compares frequency responses, and then finally returns to the original IIR Filter Construction menu to compare the quantized response against the initial filter specifications. It is assumed that the DEVELOPMENT DSP field in the System Configuration menu selection has been set to 'none', indicating non-specific, DSP chip-independent code generation. First, enter the following IIR Filter Construction menu:

FILENAME:	elliptic
DESIGN (S/B/BES/C1/C2/E):	E
SAMPLING FREQUENCY (Hz):	50000
CENTER FREQUENCY (Hz):	15000
BANDWIDTH (Hz):	10000
TRANSITION BANDWIDTH1 (Hz):	400
TRANSITION BANDWIDTH2 (Hz):	400
STOPBAND ATTENUATION (dB):	45
ORDER (STEIGLITZ ONLY):	6
RIPPLE (dB),STEIGLITZ ERR:	2

Once inside the graphical filter editor, you should see the tolerance regions displayed for the specified elliptic filter. Press the 'C' key to begin construction of the filter. Answer the "switch designs" prompt also with a 'C' to continue with the same design method. Once the IIR Filter Construction function is complete, enter the following FFT Generation menu:

FILENAME:	elliptic
FRAMESIZE:	(use init value)
FFT ORDER:	(use init value)
% FRAMESIZE OVERLAP:	0
WINDOW (R/Hm/Hn/Bl/Bt):	R
REAL/IMAG OUTPUT (Y/N):	N
PRECISION (I/R):	R
PHASE/MAG APPROX (Y/N):	N

Note the use of no OVERLAP and a rectangular WINDOW in the above example; this is standard procedure for frequency transformation of filter impulse responses. Now, enter the following Code Generate/Quantize menu:

FILENAME:	elliptic
IIR STRUCTURE (S1/S2/S3):	S3
LANGUAGE (ASM/C):	ASM
QUANTIZATION (Q8-Q31):	Q14
COEFF BASE ADDRESS / MODE:	0
ASM. CODE ROUNDING (Y/N):	N
SIMIMULATE QUANTIZED h[n]:	Y,0.25

No actual source code will be generated for this example, unless the the DEVELOPMENT DSP field in the System Configuration menu selection is set to a specific DSP device. However, upon completion of the example, a simulated quantized impulse response will have been generated as a time domain waveform. This waveform can be accessed with a filename made up of the Code Generate/Quantize FILENAME field entry, and a "_Q" appended—in the example above, the filename will be "ellipt_Q". If you now invoke the Waveform Display/Edit function, the FILENAME1 and FILENAME2 fields will already be initialized for simulated quantized and original impulse response comparison viewing:

FILENAME1:	elliptic
FILENAME2:	ellipt_Q
FRAMESIZE:	(use init value)
LOG AMP:	N
Y-LABELS (SCALE,UNITS):	1
FORMAT,PLOT:	O,C
GRID:	I

The above menu entry specifies overlaid-trace display. Depending on filter type and quantization value, you may or may not be able to detect differences in the simulated quantized and original impulse responses in the time domain. If the DUAL AUTOSCALE field in the User Setup area is at its default setting ('FILE'), then you will be able to fully compare shape, ringing,

and delay characteristics between the quantized and non-quantized impulse responses. However, you must set the DUAL AUTOSCALE field to 'OFF' to see gain effects. This is because the actual time domain output of quantized filter responses remains scaled to the quantization and input level used in the original simulation; thus, output values are exactly as they would be on the fixed-point device being simulated. To make exact measurements of gain effects due to quantization, perform FFT Generation and compare the results in the frequency domain. In all frequency domain displays, Hypersignal automatically accounts for the QUANTIZATION and INP (level) entries in the original Code Generate/Quantize function menu. If gain comparison must be made in the time domain, then you must set the DUAL AUTOSCALE field in the User Setup function to 'OFF', and perform a manual "normalization" calculation, illustrated by the following examples:

For an INP value of 0.99997, with Q15 quantization, no normalization is necessary.

For an INP value of 0.99997, with Q14 quantization, the time domain response will show a factor of 2 decrease.

For an INP value of 0.99997, with Q16 quantization, the time domain response will show a factor of 2 increase.

Thus, the true "normalized" level can always be found by multiplying the quantized time domain impulse response by:

$[2^{Q15-QN}]/INP$

where N is the quantization value, and INP the input level value, respectively, used in the original quantization simulation (e.g. CODE GENEREATE/QUANTIZE function). As a combined example, an INP value of 0.25 with Q17 quantization would require no normalization, because

$[2^{15-17}]/0.25 = 0.25/0.25 = 1$

As mentioned above, when displaying the *frequency* response of quantized filter output, no normalization is necessary, because Hypersignal-Macro automatically applies the above scaling process, producing a correction offset in dB that is added to the frequency domain data. This allows simulated quantized filter responses using various quantization and input level values to be easily compared with original design specifications in the Filter Construction graphical displays, as well as in the Magnitude and Phase Display functions. To enable this frequency domain investigation of the effects of quantization, transformation into the frequency domain is the next step. Enter the following FFT Generation menu:

FILENAME:	ellipt_Q
FRAMESIZE:	(use init value)
FFT ORDER:	(use init value)
% FRAMESIZE OVERLAP:	0
WINDOW (R/Hm/Hn/Bl/Bt):	R
REAL/IMAG OUTPUT (Y/N):	N
PRECISION (I/R):	R
PHASE/MAG APPROX (Y/N):	N

After completion of the FFT processing specified above, the frequency response of the simulated quantized impulse response waveform file, "ellipt_Q.FRQ", will be available for comparison viewing with the non-quantized frequency response created earlier "elliptic.FRQ" using the Magnitude Display function, as shown below:

FILENAME1:	ellipt_Q
FILENAME2:	elliptic
POWER SPEC:	M
AMP SQUARE (Y/N):	N
LOG (N/+/-,N/base,Min):	+
Y-LABELS (SCALE,UNITS):	1
FORMAT,PLOT:	D,C
GRID:	I

Again, overlaid-trace display has been specified. Note the use of positive log display ('+' entry), also standard procedure for filter frequency responses. Next, further measurement of the effects of quantization may be made against the original filter design specifications. To do this, return to the IIR Filter Construction menu selection. Ensure the original "elliptic" filter specifications are brought up, as follows:

FILENAME:	elliptic
DESIGN (S/B/BES/C1/C2/E):	E
SAMPLING FREQUENCY (Hz):	50000
CENTER FREQUENCY (Hz):	15000
BANDWIDTH (Hz):	10000
TRANSITION BANDWIDTH1 (Hz):	400
TRANSITION BANDWIDTH2 (Hz):	400

```
STOPBAND ATTENUATION (dB):   45
      ORDER (STEIGLITZ ONLY):   6
RIPPLE (dB),STEIGLITZ ERR:   2
```

Now change the FILENAME *field to 'ellipt_Q', and press <ctrl-Enter> to advance to the graphical filter editor*. Press the 'E' or 'M' keys to enable edit/measurement menu commands. The quantized frequency response should be displayed as an underlay—note the "Desired |H(ejw)| " and "Prev |H(ejw)|" legend at the top of the display. If either the design goal or the quantized response are not displayed, press the 'O' key consecutively until both the design goal and quantized frequency response are displayed simultaneously.

Note the areas where the response deviates from the specified design goal tolerance regions. With quantization values increasingly greater than the 14 bits chosen in the above example, deviations would become less significant. Use the 'PgUp' key to zoom in and make fine measurements on the quantized filter passband. Current cursor values in frequency (Hz) and normalized amplitude (dB, if in log amplitude display) are continuously updated at lower right of the display. Use the arrow keys to move the crosshair cursor around.

Remember that for different DSP chips and different values of quantization the results may change dramatically. Simulation of quantization effects is done as faithfully as possible for individual DSP chips, taking into account generated source code settings such as saturation mode, overflow mode, etc., parameters specified in the Code Generate/Quantize menu such as rounding, and chip design features such as accumulator length and built-in rounding algorithms.

4.10 Creating Macros

This section assumes that you have a working knowledge of individual macro language programming constructs and the syntax of macro files. For complete detail in this area, please see chapter 7, THE HYPERSIGNAL MACRO LANGUAGE. As opposed to discussing the macro language in detail, this section is intended to dicuss in general the philosophy and approach to creating useful Hypersignal macros. Special emphasis is placed on making macros "reusable" through parameter passing from the Macro Manager menu, and also through extensive use of procedures inside your macro files.

4.10.1 Writing Macros

Writing a Hypersignal macro is more or less comparable to writing a watered-down Pascal program. The macro language is much simpler (and much less powerful), but nonetheless requires you to be aware of variables (both their "types" and whether they have been initialized), conditional statements and the resulting "paths" the code might or might not take, loops, and difference between "compile-time" and "run-time". Learning to write a macro language program is not very difficult—basically, the object is combine a series of Hypersignal DSP and/or display functions in some combination. In all likelihood, the combination is one that you've already simulated more or less manually through normal use of Hypersignal. After learning a few conventions and some syntax, your macro will be running smoothly without much effort. However, through a small amount of planning ahead-of-time, you can write your macros so that they can be used to help you construct a "library" of new functions that can be used as building blocks for your current and future projects.

The Hypersignal macro language has been carefully designed to allow and encourage the creation of "reusable macros". The idea behind reusable macros is that they are self-standing, taking variable input and performing a task in a consistent, reliable manner. In this way, multiple reusable macros can be combined in arbitrary sequences as the project dictates and produce dependable results with a short amount of development time.

4.10.1.1 Reusable Macros and Reduced Architecture

The concept of reusable macros in Hypersignal is based on the assumption that the built-in menu functions in Hypersignal are distinct (from one another), powerful and flexible, and cover a very large amount of "DSP function space". By combining these built-in functions in arbitrary fashion, completely new DSP functions can be achieved. As a side benefit, these functions are also quite robust and perform a tremendous amount of error-checking—a result of their evolution in "menu driven mode", in which the user *interacts with these functions as they process data*. In many block diagram packages and "script file" packages, the individual functions may be weak and limited—because their designers were never faced with the prospect of a user actually pulling the function up on a menu, attempting to control it completely through many input parameters, and wanting to know exactly what error(s) might be occurring. For example, one of these packages might allow you to implement a delay line by invok-

ing a series of “Z-shift” blocks (or function calls). In Hypersignal, you would invoke a macro call to the Difference Equation function, where your equation might look like “y[n]=x[n–M]” (M being the length of the delay line). The use of a single, flexible function like Difference Equations, which can be used for many purposes, is a direct result of the requirement that the function perform reliably under the harsh spotlight of “user glare”.

As a result of its earned popularity and success in the “free-form lab bench mode”, Hypersignal now posseses a powerful, “reduced architecture” that allows it to cover wide DSP function and interactive display ground with about 25 basic functions. By writing reusable macros to combine these functions, it is straightforward to achieve practical DSP processing for a broad range of DSP-based applications.

The practical implementation of reusable macros in Hypersignal is based upon three constructs:

1) Passing input parameters from the Macro Manager menu

2) Procedures

3) Include files

4.10.1.2 Parameter Passing From the Macro Manager Menu

Up to input 6 parameters can be passed to a macro from the Macro Manager menu (invoked from the main menu by the ‘F3’ key). These input parameters are then referred to in the macro source as “#1", #2", etc. The input parameters can be used anywhere in the macro—in variable assignments or operations, as data-entry menu fields, as interactive display commands, etc—the occurrence of ”#n" is simply replaced with the current string in the corresponding Macro Manager field at run-time.

Also, the labels used in the Macro Manager menu can be customized, so that it will be easier to remember what input parameters are used in a particular macro over time. In fact, the entire Macro Manager menu is saved to an ASCII file that can be edited outside of Hypersignal. The filename for these files consists of the entry in the MACRO FILENAME field (any extension is stripped off) plus a “.MCF” filename extension. By editing a “macroname.MCF” ASCII file, you can change the menu labels for each input parameter field. Any text occurring before a “:” (colon) character is assumed to be label, any text occuring after is assumed to be input parameter. If you enter no colon, then default labels of the form “INPUT PARAMETER #n (blank=none)” will be used.

NOTE: Only the last 6 fields make use of the “:” character to separate label text from input parameter text. You cannot customize the field labels for the first 4 fields in the Macro Manager—if you try to do this, the results will be undefined.

4.10.1.3 Macro Procedures and “Include Files”

The Hypersignal macro language encourages the use of procedures. Basic procedures can be constructed which accept numeric or string variables. Passed variables are *always* assumed to passed by “reference”; that is, if the procedure changes the value of the corresponding variable in the calling sequence during the course of its execution, then the variable passed to the procedure will be changed upon completion. Changing the value of passed variables can be avoided if necessary by simply writing the procedure to not change any variables that shouldn’t be changed (use temporary variables inside the procedure).

Of course, the basic idea behind macro language procedures is to further extend the concept of “reusability”. By placing often-used sequences of macro language source into procedures, you achieve three advantages:

1) Less source code used. This makes the source more readable and understandable by yourself and others. Also, less source *and* object code space is required from your overall system disk space.

2) Greater reliability. Debugged and proven procedures can be “snapped in” where needed without as much concern for whether they will work properly. Less time may be spent debugging your source.

3) Further increase in the likelihood that your overall macro will become “reusable”; that is, not needing change when you decide to run the macro with different input data.

As another aide to achieving “reusability”, the Hypersignal macro language supports the use of “include files”. These are macro source code files (default filename extension of “.MAC”) that can be “included” into your overall macro source code file at any-

time. Thus, if a procedure performs a certain function for one macro, and you would like to use the same procedure for another macro, then you could place it in an include file. For example, you might specify a

{INC fftgen.mac}

line anytime you need an FFT function in your macros. Inside the "FFTGEN.MAC" file could then be a procedure that invokes the Hypersignal FFT Generation function by using the parameters passed in the procedure as input to the data-entry menu fields. Conceivably, you could achieve a very flexible FFT procedure that you could call from anywhere, controlling framesize, overlap, etc. as needed. An explicit drive and subdirectory can be specified in the include statement also, for example:

{INC C:\MACROS\filename.mac}

Otherwise, the DRIVE and SUBDIRECTORY fields currently entered in the Macro Manager menu are assumed. If no filename extension is entered, the program assumes a '.MAC' extension. If the macro filename truly has no extension, you should enter a '.' after the filename.

4.10.2 Using the "Record" Mode to Create Macros

Hypersignal-Macro contains a "Macro Record" menu function that allows you to set up and prepare a macro source file to be generated automatically as you step through a sequence of functions and displays manually. The 'F4' key from the main menu invokes the Macro Record menu, in which you can specify a macro source filename and enable the record mode. Entering the last field in this menu (or pressing the 'ctrl-Enter' key sequence regardless of editor cursor position) returns to the main menu with record mode enabled. Pressing the 'Esc' key leaves the Macro Record menu without record mode enabled. When record mode is enabled, the System Status area will show the current mode as "Macro Record Enabled", and all functions, displays, and interactive display commands will be recorded to the macro source file.

NOTE: Pressing the 'F4' key again when record mode is enabled turns off the record mode. If you forget to do this, you will eventually create a monster macro file, which may not be what you wanted.

Following are some important notes about the macro record mode:

1) Only actual functions and displays (and their data-entry menu parameters) and interactive display commands are recorded. Main menu activity, on-line Help activity (unless performed from within an interactive display), and data-entry editor activity is NOT recorded.

2) Delays inside interactive displays can be achieved by pressing the '/' key, waiting for the desired amount of time, and pressing the '/' key again. *'/' commands should be entered in pairs, with no other interactive commands entered in between.* The '/' key is only active when the record mode is enabled, and applies to all interactive waveform displays, including graphical filter design and instrumentation displays. The approximate time that you allow to elapse between the '/' pair entry will be the delay that appears when the macro is "replayed". The '/' key will elicit a beep if the SOUND field in the User Setup area is enabled, but you should ignore this.

3) The record mode can also be entered from inside a macro by a "Macro Record" statement, followed by any menu parameters and a standard "endfunc" statement. You can use an "F4" or "$F4" statement to exit record mode from inside a macro. This type of operation is not recommended, and its ultimate purpose is not clear. You should definitely not attempt to enter the Macro Record menu twice from within a macro without leaving record mode first.

4) On-line Help is available as usual from within the Macro Record menu by pressing the 'F1' key.

NOTE: Using interactive display commands extensively while the macro record mode is enabled will likely create very long macro files. This is because cursor movement inside the displays can easily add excessive interactive commands (e.g. $RIGHT, $LEFT, $UP, etc.) to the macro file being recorded. It is highly suggested that you hand-edit such macro files to reduce the number of successive interactive commands of the same type before executing the macro. This can be done using the $REPEAT(N) command, where N specifies the number of times to repeat the next command. See section 7.3.2, Issuing Interactive Display Commands From Macros, for more information about this command. Please note that currently the macro language has a limit of about 450 total macro commands, including interactive commands and menu field entries.

this page intentionally left blank

5.0 HYPERSIGNAL WAVEFORM FILE FORMAT

In order to facilitate automatic control over the various parameters and values which must be maintained during the course of waveform manipulation, Hypersignal utilizes a pre-defined waveform file "header" format. The file header allows waveform parameters to be "attached" to the waveform data they represent, and be carried along with the actual waveform file data.

Hypersignal waveform files are stored in binary format, in order to achieve mimimum disk space usage and maximum transfer rates when creating waveform files (for example, when acquiring them in real-time from DSP/Acquisition hardware). In the Hypersignal file format, the header can be a "standard" header or "extended" header. Immediately after the header must follow the waveform data, organized as continuous samples until the end of the file.

DETAILED FORMAT DESCRIPTION

Following is an explanation of the organization of Hypersignal waveform files, including a description of each header element. Noted along with the description of each each header element are is the "word index" (starting at zero; a word=16 bits), and the following "attributes":

1) Size of header element, in words (word = 16 bits).

2) A "T" and/or "F" indicator, intended to be helpful if you are creating your own waveform files. "T" means that element is *required to be non-zero* for time domain files or the results of time domain function usage will be undefined, "F" means the same for frequency domain files.

3) An "S" or "E" header length indicator. "S" indicates that element is part of standard header, "E" indicates element is also part of extended header.

NAME	WORD INDEX	ATTRI-BUTES	DESCRIPTION
MAX AMP-LITUDE	0	1,T,F,S,E	Maximum absolute value of waveform data. Value ranges from 0 to 65535.
FRAMESIZE	1	1,T,F,S,E	Length of time domain analysis framesize in samples. This parameter is carried by the FFT Generation function into frequency domain waveform files unchanged. Value ranges from 0 to 65535.
SAMPLING FREQUENCY RE-MAINDER (SFrem)	2	1,T,F,S,E	Remainder of double-word sampling frequency value (in Hertz). Value ranges from 0 to 32766. Total SAMPLING FREQUENCY (SF) can be calculated as: SF = 32767 * SFquo + SFrem where "SFquo" is word index 8, below.
FFT ORDER	3	1,F,S,E	The lower 8 bits of this word contain the FFT order, used to calculate FFT length as 2FFT ORDER. For most time domain waveform data files, this parameter starts off as zero, which disables automatic initialization of the FFT ORDER field in the FFT Generation menu selection. The File Acquisition and Analog Conversion functions give you the option of initializing this field. The Difference Equations function will initialize this field if x1 or x2 external input is used in the equation. Other processing function create output waveforms that inherit this value from the input waveforms used. **Bits 15-8: AMPLITUDE EXPONENT (AE)** In versions 3.30d and later of real-time software and 4.30d and later of non-real-time software, bits 15-8 (upper 8 bits) of this value contain the "amplitude exponent" for the waveform. In earlier versions, these 8 bits were not used and were reserved (see Word Index 6 for a description of the amplitude exponent in earlier versions). In later versions, when this value is non-zero, the maximum amplitude of the waveform can be calculated as 2^{AE*}(MAX AMPLITUDE). Thus, when AE=0, the MAX AMPLITUDE element above, and all waveform data values, are simply multiplied by 1. When used as a time domain exponent, AE ranges from -128 to 127. A time domain waveform exponent is written and used by all Time Domain functions. The File Acquisition function can also create a waveform with an exponent, upon permission from the user when data "over-ranges". The Analog Conversion function also takes

			waveform header exponents into account when sending waveform data to the D/A conversion hardware, setting the digital scaling factor used by the hardware's DSP chip to the appropriate scaling factor. When used as a frequency domain exponent, AE also ranges from -128 to 127. A frequency domain exponent is written only by the FFT Generation function, and *used by all frequency domain functions* (except Pole-Zero Plot), and also by the Filter Construction functions when graphically displaying previously calculated filter responses. Note that the "Headitor" stand-alone program can be used to examine waveform exponents (see section 5.1, Using the "Headitor" External Program).
NUMBER OF DATA VALUES REMAINDER (NDVrem)	4	1,T,F,S,E	Remainder of double-word NDV (in samples). Value ranges from 0 to 16383. Total NUMBER OF DATA VALUES (NDV) can be calculated as: NDV = 16384 * NDVquo + NDVrem where "NDVquo" is word index 9, below.
FRAME OVERLAP	5	1,T,F,S,E	Amount of frame overlap in samples. Ranges from 0 to 4095. This parameter is stored only by the LPC Autocorrelation and FFT Generation functions. When this value be non-zero, the actual, or "effective", time domain framesize used by frequency domain functions to determine the amount of time represented by each FFT-output frame can be calculated as FRAMESIZE-FRAME OVERLAP.
DATA TYPE ATTRIBUTE	6	1,T,F,S,E	The "data type attribute" contains various information about the waveform, including its time domain source, whether the waveform data is complex, whether the header includes extended information, LPC Autocorrelation order, time domain window type, and exponent scaling factor. Bits 15-0 (bit 15 being msb, bit 0 being lsb) are defined as follows: **Bits 15-12: AMPLITUDE EXPONENT (AE)** In versions 3.30c and earlier of real-time software and 4.30c and earlier of non-real-time software, this value is used as a "amplitude exponent" for the waveform. In later versions, these 4 bits are reserved (see Word Index 3 for a description of the amplitude exponent in later versions). In the earlier versions, when this value is non-zero, the maximum amplitude of the waveform can be calculated as 2^{AE*}(MAX AMPLITUDE). Thus, when AE=0, the MAX AMPLITUDE element above, and all waveform data values, are simply multiplied by 1. When used as a time domain exponent, AE ranges from -8 to 7. A time domain waveform exponent is written and used by all Time Domain functions. The File Acquisition function can also create a waveform with an exponent, upon permission from the user when data "over-ranges". The Analog Conversion function also takes waveform header exponents into account when sending waveform data to the D/A conversion hardware, setting the digital scaling factor used by the hardware's DSP chip to the appropriate scaling factor. When used as a frequency domain exponent, AE ranges from -4 to 11. A frequency domain exponent is written only by the FFT Generation function, and *used by all frequency domain functions* (except Pole-Zero Plot), and also by the Filter Construction functions when graphically displaying previously calculated filter responses. Note that the "Headitor" stand-alone program can be used to examine waveform exponents (see section 5.1, Using the "Headitor" External Program). **Bits 11-9: WINDOW TYPE** The WINDOW TYPE is written by the FFT Generation or LPC Autocorrelation functions according to the time domain window applied during processing performed by these functions. WINDOW TYPE values are as follows: 000 rectangular window (i.e. none) 001 Hamming window 010 Hanning window 011 Blackman window 100 Bartlett window 101 user-defined 110 user-defined 111 user-defined **Bits 8-4: LPC ORDER** The LPC Autocorrelation function uses bits 8-4 to store the LPC ORDER of its output waveform. Value ranges from 1-31. Note that for an LPC Autocorrelation output waveform, the value of FRAMESIZE above generally equals LPC ORDER+1. **Bit 3: COMPLEX ATTRIBUTE**

			In the time domain, bit 3 = 1 indicates that the waveform data should be interpreted as complex; bit 3 = 0 indicates "real" (non-complex) time domain data. In the frequency domain, bit 3 = 1 indicates that full-length FFT frames have been stored in the waveform, and that double-sided spectrums should be processed and displayed; bit 3 = 0 indicates single-sided spectrums should be processed and displayed. **Bit 2: EXTENDED HEADER ATTRIBUTE** Bit 2 = 1 indicates an "extended waveform header", bit 2 = 0 indicates a standard waveform header. Any Hypersignal waveform file can have either a standard or extended waveform header. An extended header adds another 54 words to the standard 10, for a total header length of 64 words, or 128 bytes. The last 32 bytes of an extended header are never used by Hypersignal, and may be user-defined. **Bits 1-0: TIME DOMAIN SOURCE** When time domain data is first acquired by the File Acquisition, Analog Conversion, and instrumentation functions, TIME DOMAIN SOURCE is set to zero. The TIME DOMAIN SOURCE is written as a non-zero value by the Filter Construction and LPC Autocorrelation functions. In most processing functions, this attribute inherits its value from the input waveforms used in the function. TIME DOMAIN SOURCE values are as follows: 00 arbitrary time data 01 LPC autocorrelation data 10 FIR or IIR filter or other transfer function ("H[n]") data 11 reserved for Signalogic use
NUMBER OF CHANNELS	7	1,F,S,E	The lower 6 bits (0-5) of this element are used to indicate the number of channels stored in the waveform data. Values can range from 0 to 63 (one channel to 64 channels). Bit 6 (NYQUIST BIT) is used to indicate whether the "Nyquist" (FFT length + 1) magnitude and phase points have been calculated by the FFT Generation function (0=no, versions before 2.0; 1=yes, versions 2.0 and later). Bit 7 (AUTOSCALE BIT) is used to indicate whether the waveform should be autoscaled when displayed with another waveform in a dual display (0=no, 1=yes). This bit is active when the DUAL AUTOSCALE field in the User Setup menu is set to 'FILE'; this bit is inactive (overridden) when the DUAL AUTOSCALE field is set to 'ON' or 'OFF'. Currently no Hypersignal function sets this bit. Bits 8-15 are available for user-defined purposes.
SAMPLING FREQUENCY QUOTIENT (SFquo)	8	1,S,E	Quotient of double-word sampling frequency value (in Hertz). Value ranges from 0 to 32767. Total SAMPLING FREQUENCY (SF) can be calculated as: SF = 32767 * SFquo + SFrem where "SFrem" is word index 2, above.
NUMBER OF DATA VALUES QUOTIENT (NDVquo)	9	1,S,E	Quotient of double-word NDV (in samples). Value ranges from 0 to 32767. Total NUMBER OF DATA VALUES (NDV) can be calculated as: NDV = 16384 * NDVquo + NDVrem where "NDVrem" is word index 4, above.
MIN AMP-LITUDE	10	1,E	Minimum absolute value of waveform data. Value ranges from 0 to 32767. This value is currently stored only by the FFT Generation function, and used only by frequency domain displays when displaying negative log data. This value is set to zero for standard header waveforms.
CALIBRATION OFFSET	11	2,E	The total offset (in dB) from the "nomimal dB peak" value that existed due to Spectrum Analyzer calibration when the waveform was acquired using the Analog Conversion function. The nominal dB peak value is defined as 0 dB, or 32767 linear (16-bit full-scale A/D input in the Spectrum Analyzer). Thus, when the ENGINEERING UNITS command was executed, the "peak" value for waveforms was re-assigned, resulting in some offset in dB from the nomimal peak value of 0 dB. Also, the user assigned some arbitrary value of dB to this value. These two values, the actual dB shift from 0 dB, and the user-entered number, are added together and stored as CALIBRATION OFFSET dB. The value is stored as an IEEE-format 32-bit floating-point value, and is stored as zero by the Analog Conversion function until the

			system has been calibrated using the Spectrum Analyzer function. This value is set to zero for standard header waveforms.
GAIN1	13	2,E	The hardware gain (in dB) that was in effect for channel 0 when the waveform was acquired using the Analog Conversion function. For most hardware, this value will be initialized to zero. For some hardware with programmable gain, this value will be set to the gain value currently programmed for channel 0. This value is set to 0 for standard header waveforms.
GAIN2	15	2,E	The hardware gain (in dB) that was in effect for channel 1 when the waveform was acquired using the Analog Conversion function. For most hardware, this value will be initialized to zero. For some hardware with programmable gain, this value will be set to the gain value currently programmed for channel 1. This value is set to 0 for standard header waveforms.
QUANT	17	1,E	An integer reflecting the QUANTIZATION field value (in bits) that was in effect if the waveform was generated by the Code Generate/Quantize function. Zero otherwise. This value is used by frequency domain displays to allow convenient comparison of quantized filter impulse responses with their non-quantized counterparts. See section 4.9.5.1, Simulating the Effects of Quantization, for more detailed information.
INP LEVEL	18	2,E	A 32-bit floating-point value reflecting the INP level (the actual INP subfield value multiplied by $2^{QUANTIZATION}$) that was in effect if the waveform was generated by the Code Generate/Quantize function. Zero otherwise. This value is used by frequency domain displays to allow convenient comparison of quantized filter impulse responses with their non-quantized counterparts. See section 4.9.5.1, Simulating the Effects of Quantization, for more detailed information.
START FREQ	20	2,E	A 32-bit floating-point value reflecting a starting frequency value in Hertz for frequency domain waveforms where the first value in each frame does not represent zero (dc). This value is used by frequency domain displays to depict a segment of bandwidth, where the FFT processing reflects a starting point in the waveform spectrum not at zero frequency.
RESERVED	22-31	10,E	This extended header space is reserved for Signalogic. In order to maintain compatibility with future versions of Hypersignal, it is suggested that you not use this area for your own purposes. If a waveform is being created from scratch, this area should be written with zeros. If a waveform is being created from an existing waveform, this area should be transferred verbatim.
USER-DEFINED	32-63	32,E	This extended header space is available for user-defined purposes, and will not be used by Signalogic.
WAVEFORM DATA VALUE	M	1,T,F,S,E	First data value in waveform.
WAVEFORM DATA VALUE	M+1	1,S,E	Second data value in waveform.
WAVEFORM DATA VALUE	M+2	1,S,E	Third data value in waveform.
.			
.			
WAVEFORM DATA VALUE NDV–1	M+NDV –1	1,S,E	NDVth data value in waveform.

NOTE: The starting waveform data value word index M, above, equals 10 for standard header waveforms, and 64 for extended header waveforms. In terms of ASCII values (for example, when considering the output of the ASCII export function), a Hypersignal waveform has 10 values in a standard header, and 60 values in an extended header. Note that there are 60 ASCII values instead of 64 because 4 values in the header, as it is currently defined, are floating-point, requiring 2 words each. This interpretation might change in the future, or due to user-definition of the last 16 words of extended headers.

INTERPRETATION OF WAVEFORM DATA

All Hypersignal waveform files conform to the same file format; however, the way the data is organized in the file can be interpreted by Hypersignal differently, depending on two factors:

1) filename extension

2) COMPLEX ATTRIBUTE and NUMBER OF CHANNELS values

Waveform files in Hypersignal fall into two categories, based on their filename extension: time domain and frequency domain. The major difference in the way Hypersignal interprets waveform file data results from these two types of of data.

TIME DOMAIN WAVEFORM FILES

Time domain waveform files are usually created by the File Acquisition and Analog Conversion functions, although there are many other ways to generate them, including Difference Equations and Filter Construction. They can also be generated externally to Hypersignal. See section 3.8, Creating Hypersignal Waveforms Files, for more information.

Time domain data consists of periodically sampled sequences of values that represent "real-world", continuous, analog waveforms. Time domain waveform files must always have ".TIM" filename extensions.

For time domain files, the data is assumed to be in a "sample-based" format. Each sample follows immediately after the previous to form a linear sequence of samples. Two parameters in the waveform file header can cause an enhancement of this interpretation: the COMPLEX ATTRIBUTE, and the NUMBER OF CHANNELS value.

If the complex attribute is set in the waveform header, each sample becomes a "complex sample", with real component stored first, and imaginary component second. Note that the NDV value above *always represents the total number of values—not samples*; thus, the number of complex samples in a complex waveform file = NDV/2.

If the waveform is multichannel (NUMBER OF CHANNELS element in waveform header > 0), then Hypersignal interprets the data as *interleaved*. That is, the first sample is assumed to be from channel 0, the second from channel 1, and sample K-1 from channel K-1, where the total number of channels = K. This sequence then repeats in groups of K samples, with the next sample (sample K) being assigned to channel 0, and so on. Thus, the number of samples in a multichannel waveform file = NDV/K. Note that if the data is also complex, then the sequence repeats in groups of every 2*K actual *data values*; in this case the number of complex samples in the file would be NDV/K/2.

FREQUENCY DOMAIN WAVEFORM FILES

Frequency domain waveforms are usually generated by the FFT Generation, Power Spectra Generation, and Transfer Function Generation functions. They can also be generated by the Difference Equations function and externally, outside of Hypersignal. See section 3.8, Creating Hypersignal Waveforms Files, for more information.

Frequency domain waveform files have ".REA" (real component), ".IMA" (imaginary component), ".FRQ" (magnitude), ".PHZ" (phase), and ".PWR" (power spectra) filename extensions. Real and imaginary component waveform files represent the real and imaginary output of the FFT, *and are created by Hypersignal only if the REAL/IMAG OUTPUT field in the FFT Generation function was enabled.* Magnitude waveform files represent the squareroot of the sum of the squares of the real and imaginary FFT output data. Phase waveform files represent the arctangent of the imaginary data divided by the real data and then multiplied by 4096, or 4096∗ARCTAN(imag/real). The FFT Generation function always stores magnitude and phase waveform files as part of its output. Power spectra files are created by Hypersignal as output from the Power Spectra Generation function, and represent frame-based averages of the correponding magnitude waveform file.

For frequency domain files, the data is assumed to be in a "frame-based" format. Real and imaginary waveform files always use a frame length (FL) equal to N, where N = FFT length, or $2^{\text{FFT ORDER}}$. When the original time domain data was not complex, all other frequency domain files use a frame length of FL = N/2+1. Each FFT-output frame, consisting of FL samples follows immediately after the previous to form a linear sequence of frames that represent the frequency transformation of the original time domain waveform. In this basic case, NDV = the number of frames ∗ FL. Two parameters in the waveform file header can cause an enhancement of this interpretation: the COMPLEX ATTRIBUTE, and the NUMBER OF CHANNELS value.

If the complex attribute is set in the waveform header, then FL is always equal to the FFT length, for both real/imaginary waveforms and all other types. Since the original time domain data was complex, it can no longer be assumed that the real and imaginary FFT output data was "symmetrical" about the N/2+1 point, forcing all FFT output data to be stored. Thus, in this case, all

frequency domain functions process and display “double-sided” spectrums. For double-sided frequency domain data, NDV = the number of frames ∗ N.

If the waveform is multichannel (NUMBER OF CHANNELS element in waveform header > 0), then Hypersignal interprets the data as *frame-interleaved*. That is, the first frame is assumed to be from channel 0, the second from channel 1, and frame K-1 from channel K-1, where the total number of channels = K. This sequence then repeats in groups of K frames, with the next frame (frame K) being assigned to channel 0, and so on. Thus, the number of frames in a multichannel waveform file = NDV/K/FL. Note that if the original time domain data was also complex, then the number of frames in the file would be NDV/K/N.

Note that all waveform data values can range from 2.93873e-39 to 5.575e+42 in version 3.30d and later (4.30d and later for non-real-time software) due to the block floating-point waveform storage method used for all Hypersignal waveforms. The range is from 3.90625e-03 to 4.194176e+06 for versions before 3.30d (and 4.30d for non-real-time software). Waveform data values are stored as 16-bit “mantissas” in the data segment of the waveform file, along with an overall waveform “exponent” stored in the header segment of the file. This scheme provides both extremely fast throughput, especially for real-time functions such as Analog Conversion, where disk streaming speed is critical, and fairly good dynamic range. Use of this scheme means that input data may still need to be scaled if its range is too large or not sufficiently large for the above range. The File Acquisition function will perform data ranging automatically, after obtaining permission from a menu prompt. If you are acquiring waveform files that still exceed the allowed range, then you may need to some arbitrary scaling in an external program prior to using the File Acquisition function. For instance, if you are generating floating-point waveform file data ranging from -0.00001 to 0.00001, the data should be scaled up prior to Hypersignal processing.

If you are generating waveform file data in an ASCII format, then each data value (and header element if you intend to use the ASCII Import function instead of the File Acquisition function) should be on a separate line. Thus, all values should be arranged in a single, contiguous column. Much greater flexibility in ASCII file translation is possible with the File Acquisition utility—see section 4.1.1, Acquiring Waveforms From Disk Files.

5.1 Using the “Headitor” External Program

The “Headitor” is a stand-alone program included on distribution diskette #2, intended to allow parameter-by-parameter inspection and modification of Hypersignal waveform file headers (or, the “header editor”). Use of Headitor is mostly self-explanatory, but some basic instructions are worth noting in advance:

1) To run Headitor, type:

headitor <car ret>

2) When it runs, the headitor program will first look for its own configuration file (“HEADITOR.CNF”, not to be confused with the standard Hypersignal “HSxxxxx.CNF” file) to find an initial waveform filename. If not found, then Headitor looks for a standard Hypersignal configuration file, and if found, uses the default TIME DOMAIN FILENAME field value stored from the User Setup area as its initial filename. Alternatively, you can specify a filename in the Headitor command line, as in:

headitor filename <car ret>

If no extension is entered, then a “.TIM” extension is assumed. If no drive and/or subdirectory is specified in “filename”, then Headitor uses the DRIVE and SUBDIRECTORY field values stored from the User Setup area (assuming a standard Hypersignal configuration file can be found). If no filename can be found or is given, then Headitor will come up and display null entries for all waveform header parameters, until a filename is entered on the first line of display.

3) When Headitor runs, the first field contains the full filename specification, including drive, subdirectory, and waveform filename extension. No default values are assumed, or can be entered elsewhere. If you need to change any of these values, especially the waveform filename extension (for example, “.FRQ” for a frequency domain magnitude waveform file), then you must enter the new values in this field. After entry of a filename in this field, waveform header parameters are then displayed (or re-displayed) in the fields following the filename field.

4) The editing functions in Headitor are similar to the built-in data-entry editor in Hypersignal. See section 3.4, Using the Hypersignal Data-Entry Editor.

5) Some of the more important Headitor commands:

ESC leaves Headitor *without saving any changes you’ve made to waveform header parameters*

F1	reloads the current waveform header specified in the filename field
F10	saves current changes to the waveform header and leaves the Headitor

this page intentionally left blank

6.0 BRIEF DISCUSSION OF SAMPLED DATA SYSTEMS

One of the fastest growing areas of high technology today is the field of digital signal processing (DSP): the use of digital computers, microprocessors, specialized signal processing ICs and other digital machines to process physical signals that vary with time or space.

The rapidly increasing level of interest in digital signal processing is largely a matter of economics. The proliferation of low-cost, flexible, personal computers has created an unprecedented number of opportunities to develop and apply DSP techniques to areas that can benefit from digital signal analysis. Also, the advent of inexpensive but fast digital signal processor ICs has allowed entry into real-time product realms once ruled exclusively by analog systems, as well as into product areas formerly served less effectively by slower microprocessors, or with high data rate requirements that are only now economically feasible to develop.

The flexibility and power of digital signal processing makes it a tool well-suited for dealing with an increasing number of applications. However, no matter what the application, actual real-world signals must be in a form the digital machine can accept before any processing can take place. The form a signal must assume prior to digital signal processing is the subject of sampled-data systems. This section is not meant by any means to be a highly technical or rigorous explanation of sampling theory, but rather to present some very general concepts about sampled data acquisition systems. The interested reader is encouraged to delve into the myriad books on the subjects of sampled data systems and digital signal processing in general.

A SAMPLED SIGNAL

A sampled signal is usually meant to be a representation of some physical quantity sampled at different points in time. Sampling simply means to measure the signal at an instant in time. For example, an electric signal might be 5 Volts at one instant, and 5.1 Volts slightly later. In the real world, or "analog" world, the signal is continuous—it has a value at *every* instant in time—and, of course, is required by the laws of physics to transition continuously from 5 Volts to 5.1 Volts. In the digital world, the signal is defined only at the sample times. Thus the digital machine knows only that the signal was 5 Volts at some point in time and 5.1 Volts some interval later. Obviously, the digital machine could sample the signal faster and faster until its representation of the signal approached the analog version. However, if a digital device were somehow constructed that could sample the signal continuously, it would have no processing time left over to work on the samples it obtained! Thus, the trick is to sample fast enough to obtain a reasonable facsimile of the signal, yet slow enough to have time left over to do some real work, otherwise known as "signal processing".

When the work to be performed on the signal in order to extract or determine desired results can be done for infinitely long periods, with no loss or skipping of sample values, then the system is said to be operating in "real-time". The slowest rate at which a digital machine can sample the signal and still maintain an accurate representation of it has been the subject of much study, and is discussed briefly under "data acquisition", below. In short, the slowest rate depends on the signal itself—how abruptly it changes, forcing the digital machine to obtain new samples to stay current. Most often, sampling is performed at regular intervals—the rate of sampling is constant. This is known as "periodic sampling". Periodic sampling greatly simplifies digital analysis, as will soon be apparent (all Hypersignal waveform processing functions assume periodic sampling). The sampled signal is then usually referred to as a "discrete time sequence", where the sequence values are defined only at the sample times (the sampled signal should not be thought of as having zero value at all other points in time, however—the sampled signal is simply *undefined* between samples).

DATA ACQUISITION

Data acquisition is a term used to describe the process of "sampling" a signal (capturing a sequence of numbers for a digital machine to work with, as described above) from a real-world system. In many cases, the signal captured represents the physical quantity of electric voltage, for two reasons:

1) Most signals of another type can be readily transformed into voltage, through the use of transducers such as antennae, microphones, thermistors, light-sensitive diodes, piezoelectric material, etc.

2) There exist many inexpensive devices that perform periodic time-sampling of voltage and are compatible with existing integrated circuit technology.

In the field of digital signal processing, the transformation from a continuous analog signal to a discrete sequence is usually referred to as analog-to-digital conversion (A/D conversion). There currently exists a wide variety of A/D systems for use with the IBM PC/XT/AT and compatible machines. Usually these systems include both a plug-in circuit card and software, such as Hypersignal-Macro Real-Time, that allows convenient data acquisition and storage on disk for subsequent processing and analysis.

In the case of Hypersignal software, various real-time processing options are also available, allowing signal modification or information extraction to be performed *prior* to display or disk storage.

In order to set up a data acquisition system it is important to know some information about the signal to be captured. The most important characteristic is the bandwidth of the signal. The bandwidth of a signal refers to the difference between the highest frequency at which the signal can change, and the lowest. Since many signals may not change at all for some length of time (zero frequency is sometimes known as "dc"), bandwidth often refers simply to the maximum signal frequency component. Frequency, an oft-used term in digital signal processing, may be thought of as the rate at which a signal changes values. The *maximum* signal frequency occurs when a signal changes from its lowest possible value to its highest possible value or vice versa in the smallest amount of time. In mathematical terms, 1 divided by this minimum time interval is the maximum signal frequency. Thus, frequency is expressed in units called Hertz (Hz), where 1 Hz is 1/1sec. Once the bandwidth of the signal is known, it is possible to determine the minimum rate at which the signal can be sampled (the maximum time interval between adjacent time samples) and still be truly representative of the continuous analog time signal.

One may ask what is to be gained by sampling at the minimum rate. There are two very important resources which almost always need to be conserved, processor time between samples for actual digital signal processing work, and processor memory storage for samples. The minimum rate of sampling is known as the Nyquist Frequency (Fs), and is related to the inherent bandwidth (BW) in the signal to be sampled. This relationship can be expressed mathematically by the following equation:

$$Fs>2(BW)$$

This rather famous equation states that the continuous signal must be sampled at a rate which exceeds at least twice its bandwidth in order to achieve an adequate sampled version of the signal. This relationship is of fundamental importance in setting up a data acquisition system.

ANALOG SIGNAL BANDLIMITING

A logical question might now be: how does one guarantee a limit on the bandwidth of a signal? In fact, most signals are "bandlimited" prior to analog-to-digital conversion because of the input device (or "transducer") from which they originate or the analog circuitry through which they must subsequently travel. In many cases, natural bandlimiting is in effect because the transducer being used has an inherent bandlimit—some examples of this might be a microphone that cannot pick up frequencies higher than 20 KHz, a telephone circuit that drops off sharply after 3.5 KHz simply because of the type and length of wire used to carry its signals, or a light-sensitive diode with a switching speed limited by the molecular structure and length of electron travel path in the silicon used to build it.

Sometimes, however, the A/D system used must provide additional, active bandlimiting circuitry, and even provide for an adjustable bandlimit frequency. This becomes necessary when the required sampling rate, determined using the "maximum frequency" and "real-time work" criteria discussed above, is less than the natural bandlimit of the transducer. Usually, this is accomplised using what is known as a lowpass filter (anti-aliasing filter), which "passes" frequencies below a certain "cutoff frequency" and blocks those above. Theoretically this cutoff frequency may be sharply defined, but in practice sharp cutoff frequencies are difficult to achieve. See the Hardware Reference Guide for more information on anti-aliasing filters and techniques for use with both naturally and artifically bandlimited systems.

Once you have an appropriate A/D system producing samples of a bandlimited signal at a sufficient sampling rate, you have crossed the analog-digital boundary and entered the world of digital signal processing. The information contained in the original continuous analog signal still exists, but in an equivalent form, one that digital machines and their programs, such as Hypersignal, can deal with.

this page intentionally left blank

7.0 THE HYPERSIGNAL MACRO LANGUAGE

The Hypersignal macro language can be described as a "watered-down combination of Pascal and C". As with other languages, variables and programming constructs are available to store results, isolate often-used segments of code, invoke standard mathematical, file, and I/O functions, repeat segments of code, and make decisions based on results—in general, to solve problems. Unlike other languages, the macro language provides a direct hot-line to Hypersignal functions and displays, not only allowing them to be invoked and combined in any sequence, but to accept macro language variables as input parameters. Thus, high-level language problem solving becomes married to the powerful functions and displays in Hypersignal, making them appear almost as an extensive library of callable routines.

As with most high-level languages, in the Hypersignal macro language variables must be declared to be one of several types before use, procedures can be defined before the main body of source code, mathematical operators and functions can be invoked, and various conditional and looping constructs are available. The macro language approach allows unique ability, such as gracefully transitioning from interactive mode to automatic mode. For example, the system can stop in the middle of an automated test procedure, wait for the operator to make measurements, save a display trace to disk file, set parameters, and then resume the test procedure—all without changing the current interactive function and its basic display screen.

This chapter is intended to serve as a reference for the Hypersignal Macro language. Topics covered include the way Hypersignal menu processing and display functions are controlled and invoked, the macro language syntax, all programming constructs, and all built-in functions. Also, the editing, compilation, and execution steps necessary to generate a macro from scratch, and all error messages, are listed in this chapter. For a more philosophical discussion, including techniques designed to produce clear, easy-to-use, and reusable macros, please see section 4.10, Creating Macros.

Also, please see the example macro source files provided on the Hypersignal distribution diskettes. Example source files include "DEMO.MAC", which puts on a flashy demo of Hypersignal using several waveform displays, "LOOPS.MAC", which offers a tour de force in loops and nested loops, and "SAMCONV.MAC", which is a good example of something useful: arbitrary sample rate conversion using Macro Manager menu input parameters.

7.1 Macro Source, Object, and Configuration Files

In order to create a macro, you must first create a macro source file. This file is then compiled by the built-in Hypersignal-Macro compiler, which in turn creates a macro object file. This object file is what Hypersignal actually executes. Thus, after a macro source file has been compiled once, only the macro object file need be accessed to execute the macro repeatedly. A compiler mode can be set in the Macro Manager menu that controls the circumstances under which compiling occurs. The compiler mode includes a "make mode" value which automatically compiles source files only if they have been changed. See section 7.10 below, Compiling Macros, for more information.

Whenever you set up a macro for execution in the Macro Manager menu, a third file is automatically created: the macro configuration file. The configuration file basically stores in ASCII-text form all Macro Manager menu setup and input parameter fields. Also, the labels for the input parameter fields can be edited directly in this file, allowing you to customize the Macro Manager menu for different macros. This can be especially helpful for macros with several input parameters, or when you have several macros, or when running a macro after not using it for a long time. See section 4.10.1.2, Parameter Passing From the Macro Manager Menu, for more information about the customization process.

Macro source files should normally have ".MAC" filename extensions; however, other extensions may be used by explicitly entering them in the Macro Manager menu MACRO FILENAME field when setting up the macro for compilation/execution.

Macro object files always have ".MXE" filename extensions—this extension is attached automatically by the built-in Hypersignal compiler.

Macro configuration files always have ".MCF" filename extensions—this extension is attached automatically by the Macro Manager menu function.

Macro source files are normally created in two ways:

1) by standard ASCII text-editor (manually)

2) by Hypersignal (automatically, using Macro Record function)

The first method is programming by any other name. You sit down with a text-editor, and, using the documentation and other macros as a guide, create the appropriate ".MAC" text file(s) to perform a certain function or series of functions. You may end up with a simple, short text file, or you may end up with a main text file and one or more include text files, possibly containing procedures, that make up a more complicated program.

The second method can be utilized by placing the Hypersignal software in "record mode", in which the software writes out all data field entries and interactive display commands as each function is executed from the main menu. This allows you to step through a sequence manually, and have the software generate the appropriate macro source file to then repeat this sequence later. See section 4.10.2, Using the Record Mode to Create Macros, for more information.

Conceivably, it is also possible to generate macro source files via a third means: a separate, external program. This might be useful for creating a series of interrelated macros that must execute depending on external circumstances. An example of such an application might be a graphical interface intended for test system operators, who build a test suite by selecting icons; the application software they're using then creates the appropriate macros; finally, Hypersignal-Macro executes the test suite.

7.2 Macro Language Conventions and Syntax

In the Hypersignal macro language, several basic conventions apply to all constructs in the source file. All statements must be on one line; lines can be up to 225 characters in length. Semicolons are never required at the end of a statement (or anywhere else in the language—but they can be used as the comment delimiter). In most cases, spaces or commas can be used as separators between items on the same line, such as variables being declared, or input parameters to a function, or data-entry menu fields being passed to a processing or display function.

The sections below detail the conventions and syntax used for comments, variable names, and reserved constants.

7.2.1 Macro Comments and Blank Lines

On any line, all text following a semicolon (';') is defined as comment text—hereafter simply referred to as comments. Comments can be inserted anywhere in a macro source file. The macro compiler, as one of its first actions when processing each line from a macro source file, looks for a ';' character. If if one is found, the compiler simply removes it and all text after it until the end of the line from further consideration.

The use of comments are highly recommended when writing macro source files. You may note that Signalogic-created macros provided on the distribution diskettes make extensive use of comments, especially when listing data-entry menu field parameters, which, without the benefit of their usual menu descriptions and labels (and even sometimes with), tend to be somewhat cryptic.

Any number of blank lines can be inserted into macro source files at any point. Judicious use of blank lines often improves readability and clarity of the macro source by helping to delineate a section of code pertaining to one purpose from that of another.

7.2.2 Macro Case Sensitivity

Nothing in the Hypersignal macro language is case-sensitive. All text is converted to upper case when read in from the macro source file by the macro compiler.

7.2.3 Macro Variables

Variables in the Hypersignal macro language *must be declared before use.* Variables can be one of three types:

1) Integer (32-bit integer ranging from -2147483648 to 2147483647).

2) Real (32-bit IEEE floating-point ranging from $\pm 1.5 \times 10^{-45}$ to $^{\pm}3.4 \times 10^{38}$).

3) String (up to 128 characters of ASCII text).

4) Boolean (32-bit integer with value of 0 or 1).

The Boolean variable type can be assigned the values of 'TRUE' and 'FALSE', which are intenal constants. TRUE evaluates to 1, and FALSE evaluates to 0. Boolean variables are treated for all purposes identically to integer variables—there is no difference except in declaration.

Variable names in the Hypersignal macro language can be up to 25 characters in length, and they must begin with a non-numeric character. Names longer than 25 characters can be used, but only the first 25 characters will be significant. Also, variable names cannot make use of the '#', '@', '.', '$', or other characters used by the macro language (including arithmetic operators and relational comparisons). Variables must be declared before they are used, but they can be declared at any place in the source file. For example, to declare three integer, three real, and three string variables, you could enter:

```
integer int1,int2,int3
real real1,real2,real3
string str1,str2,str3
```

or, as an equivalent declaration:

```
int1,int2,int3: integer
real1,real2,real3: integer
str1,str2,str3: string
```

Variables intended to be global, that is, intended to be used by both the main body of macro source code and one or more procedures, must be declared *before* any procedures that refer to them. Procedures can also contain local variables, that are used only by the declaring procedure—see section 7.8, Macro Procedures.

When both integer and real types are used in assignment statements, some assumptions about type-conversions are made by the macro language compiler. See section 7.5, Macro Assignment Statements, for complete information.

Up to 56 variables and procedure names can be declared in any macro source file. Variable names cannot be used that are the same as reserved constant names used by the Hypersignal macro language (see the next section).

NOTE: **Like most high-level languages, the Hypersignal macro language does not force you to initialize variables. Once a variable has been declared, the compiler makes no check as to whether the variable has appeared on the right side of an assignment statement at least once. Use of uninitialized variables will produce unpredictable results.**

7.2.4 Macro Constants

Constant values in the Hypersignal macro language fall into one of two categories:

1) User-Declared

2) Pre-Defined

User-declared constants are either numeric values or string values. They can be used in assignment statements, as input to functions, as parameters to procedures—in short, usually anywhere a variable can be used on the right side of the '=' sign in a statement.

Integer constants are formed by a "decimal" combination of the digits 0 through 9, and the '+' and '–' characters. Some examples:

0
–1234
+5678
33554432

Also, hexadecimal integer constants are allowed, using '$' followed by up to 8 hexadecimal digits:

$ABCD
$12345678
$FFFFFFFF
$8000

Real constants are formed by a "floating-point" combination of the digits 0 through 9, the '+' and '–' characters, the 'E' character, and a decimal point. Some examples:

0.0
–1234.5678
5678e10
1.2e–05
1.123456E15

Any use of the '.' character (decimal point) or 'E' character (exponent) in a numeric constant causes the constant to be interpreted as a real value.

String constants are formed by the use of double-quote marks. Some examples:

"text"
"A"
"radar12.tim"
"the brown fox jumped over the lazy dog"

String constants may contain up to 40 characters of any text besides semicolons inside the "" marks. Semicolons will be interpreted by the software as the start of comment text (see section 7.2.1, Macro Comments and Blank Lines.

In addition to user-declared constants, the Hypersignal macro language contains several built-in constants that can be referred to by pre-defined variable names. These names are reserved words—attempting to define variables with the same name will result in an error condition. Built-in constant names and their values are listed below:

Name	Value	Type
Amp	0	Integer
Fr	1	Integer
Fs	2	Integer
FFTord	3	Integer
Len	4	Integer
Overlap	5	Integer
TypeData	6	Integer
NumChan	7	Integer
Exponent	65	Integer
Off	0	Integer
On	1	Integer
File	2	Integer
Pi	3.141529...	Real

Several of the above constants are intended to be of aid when specifying waveform header parameters during use of the PutHeaderValue and GetHeaderValue functions.

7.3 Invoking Hypersignal Menu Functions From Macros

Hypersignal menu display and processing functions are invoked from a macro by simply listing the name of the function you would like to execute, exactly as it appears on the main menu. The data-entry menu fields you would like to use as input (if any) to the function should be listed starting on the next line.

For example, to invoke the Difference Equations function from a macro, you would simply enter:

```
Difference Equations
endfunc
```

as two separate lines in the macro source file. Note that 'endfunc' is always required when invoking a processing or display function. Only the spelling of the processing or function name matters; indentation, comment after the name, case of the letters are all immaterial.

Any data-entry menu fields and/or interactive commands you would like to use as input to the function should be listed after the function or display name, and before 'endfunc'. See the next two sections for several examples.

For macro purposes, one should think of Hypersignal menu functions as being divided into three categories:

1) processing functions

2) display functions

3) external programs

Once a processing function has been invoked from a macro, it will run to completion if no errors have been made in the data-entry menu fields, and if no irrecoverable errors (input file does not exist, disk full, write-protect is activated, etc.) occur during macro execution. With few exceptions, user-prompts that normally appear in manual mode are assumed to be answered one way or another in macro mode. For example, a prompt asking you whether to overwrite an existing waveform file would be assumed in macro mode to be 'Y'. As another example, a prompt that might appear in the FFT Generation function asking you whether or not to use the 80x86 and 80x87 instead of the DSP Accelerator (because, for example, the DSP board did not have enough memory for the FFT size specified) would be also be answered positively by the software. Generally, the assumption in macro mode is that you want the function to get done—any problems that might occur have already been accounted for by working through the function at least once in manual mode.

On the other hand, once a display function has been invoked from a macro, it will not finish completely until you have issued the appropriate interactive commands. With these functions you could, for example, start the display, use the '$Esc' interactive command to return to the data-entry editor, change some data-entry menu fields, and return to the display using the '$Run' interactive command. Generally, this means that a '$Esc $Esc' sequence must be used to fully stop a display function and continue execution of the macro: the first $Esc returns to the data-entry editor, the second to the main menu (however, one exception involves the use of the '$Manual' command, which temporarily returns control of the interactive display to the user). See section 7.3.2, Issuing Interactive Commands From Macros, below for a complete description of all interactive commands.

External programs can also be invoked from macros, similarly to processing functions as described above. In this case, any menu parameters, or data-entry menu fields, listed are passed as command-line parameters to the external program. Up to 10 command-line parameters can be entered after an external program invocation. Typically, an external program would be invoked from a macro by the menu label set up for it in the Ext Program Config menu. However, the following generic commands are also available, in case the name of the external program is not known at compile time, or, more likely, the Ext Program Config menu cannot be configured until run-time:

```
User-Defined Time 1
User-Defined Time 2
User-Defined Freq 1
User-Defined Freq 2
User-Defined Util 1
User-Defined Util 2
```

The generic commands are available because in some cases the Ext Program Config menu must be configured from the macro file, at run-time, or possibly several external programs are involved; thus, correct and complete names may not be known at compile time. See Tutorial section 15.6, Invoking External Programs From a Macro, for some detailed examples of this process. If you have Hypersignal-Acoustic software, see also the TEFRD.MAC file on distribution diskette #2.

7.3.1 Entering Menu Fields From Macros

Hypersignal menu display and processing functions are invoked from a macro by simply listing the name of the function you would like to execute, exactly as it appears on the main menu. The data-entry menu fields you would like to use as input (if any) to the function must be listed immediately after the function and before 'endfunc'.

For example, to invoke an FFT function, you might enter in your macro source file:

```
FFT GENERATION
    data            ; input filename
    128             ; analysis framesize = 128 samples
    9               ; FFT size = 512
    64              ; 50% overlap
    Hm              ; Hamming window
    N               ; no real/imaginary output
    R               ; real precision (this field is displayed
                    ; at run-time only if no DSP Accel-
                    ; erator installed)
    N               ; no phase/mag approximation (only
                    ; used if no DSP Accelerator
                    ; installed)

endfunc
```

Identical operation is achieved with the next example:

```
FFT GENERATION
    data 128 9 64 Hm N R N

endfunc
```

As another example, to invoke the Convolution/Correlation function, you might enter in your macro source file:

```
CONVOLUTION/CORRELATION
    speech          ; input filename
    filter          ; filter filename
    sp_out          ; output filename
    –               ; specify convolution

endfunc
```

IMPORTANT RULES FOR DATA-ENTRY MENU FIELD ENTRY

1) Data-entry menu fields cannot start on the same line as the function reference.

2) After the function reference, an 'endfunc' statement or any interactive command terminates the data-entry menu field section. Entering data-entry menu fields anywhere else will result in an error condition.

3) You may invoke a function without any data-entry menu fields. However, if the function involves an interactive display, you must still use the $Run interactive command (see section 7.3.2, below) to make the display start.

4) You can enter all or only the first few data-entry menu fields, as desired. Entry is always assumed to be sequential; thus, if you wish to enter only the 3rd data-entry menu field, you must use the 'no_change' entry for the first two fields (see below).

5) Entry of more than the maximum number of data-entry menu fields will result in an error condition. Check the function in manual mode to determine the maximum number of data fields.

6) Comments can be freely entered after or in between data-entry menu fields.

NOTE: **During normal operation, Hypersignal-Macro initializes various fields in data-entry menus automatically based on information contained in waveform headers. This usually happens when a data-entry menu is first entered, requiring the program to load a waveform specified in a waveform filename field. It can also happen when you change waveform filename field entries inside a data-entry menu. This type of operation, while extremely convenient in manual mode, poses a basic conflict with macro operation, since usually one or more data-entry menu fields are specified before the 'endfunc' state**

ments in the macro. When a function or display is invoked from a macro, all data-entry fields not marked 'no_change' or 'don't_care' (see below) take precedence over the automatic load and initialize. *However,* if, from inside the macro file, you use interactive commands (see section 7.3.2 below) to cause *another* waveform load (by causing a filename entry field to be entered), then the automatic load will take precedence, and one or more data-entry fields containing previous macro file inputs may be overwritten. Using interactive commands to enter waveform filenames must be done carefully and is only recommended when necessary.

RESERVED DATA-ENTRY MENU FIELD ENTRIES

The following reserved words can be used as data-entry menu field entries, and are interpreted with special meaning by the compiler. These words are only considered reserved between a processing or display function name and 'endfunc'.

no_change	Moves entry to next field, without changing contents of current data-entry menu field.
don't_care	Moves entry to next field, without changing contents of current data-entry menu field.
blank	Blanks current data-entry menu field.
TimeFile	Is replaced by current value of User Setup area TIME FILENAME field at run-time.
FreqFile	Is replaced by current value of User Setup area FREQ FILENAME field at run-time.

7.3.1.1 Use of the '#' and '@' Functions in Data Fields

Although the '#' and '@' functions are fully explained in section 7.10.3 below, Macro Special Functions: '#' and '@', their use in entering menu fields from macros deserves extra discussion. Use of the '@' function is especially important in this context, providing the only way to reference a variable and put it's current value to use as part of a menu field entry.

As explained in section 7.10.3, the '#' function refers to a Macro Manager input parameter, and the '@' function converts a variable or constant to a string. These functions are tremendously useful when entering data-entry menu fields. For example, suppose you wanted to acquire several waveform files from DSP/Acquisition hardware using the Analog Conversion function. By placing the Analog Conversion specification inside a nested for-loop, you could vary the filenames under which the data were stored:

```
for I=0 to num_categories
    for J=0 to num_entries
        ANALOG CONVERSION
            name@I@J    ; input filename
            Am          ; acquire/generate
            0,1         ; channel list
            1,1         ; input, output gain
            0           ; trigger channel list
            100000      ; number of samples
            20000       ; sampling frequency (Hz)
            0           ; post-acquisition framesize
            0,0         ; input, output offset
        endfunc
    end

end
```

As another example, suppose you wanted to use a Difference Equation to operate on an input waveform using two input parameters and a variable that was changing. The following example creates a series of sine waves with different offset phase values:

```
for J=1 to 100
    Difference Equations
        sine@J              ; output names: sine1, sine2, etc.
        y[n]=10000*
        sin(2*pi*n*#1/#2+@J)
        blank
        blank
        none                ; no input waveforms none
        #3                  ; output length is also input parameter
        #2                  ; sampling freq of input waveform
        #3                  ; calculate for remainder of waveform
    endfunc

end
```

In the above example, '#1' refers to input parameter #1 in the Macro Manager menu, which, according to the equation used, should specify the fundamental frequency of the sine waves. '#2' refers to sampling frequency, also an input parameter to the macro. '#3' refers to the total length (samples) desired for each output waveform. The use of '@J' causes the value of J to be converted to a string and inserted in the data-entry menu fields as appropriate. Below is the same example, but with the Difference Equation Fs constant used for the sampling frequency in the equation (in this case, Fs refers to the #2 reference in the SAMPLING FREQUENCY field):

```
for J=1 to 100
    Difference Equations
        sine@J              ; output names: sine1, sine2, etc.
        y[n]=10000*
        sin(2*pi*n*#1/Fs+@J)
        blank
        blank
        none                ; no input waveforms none
        #3                  ; output length is also input parameter
        #2                  ; sampling freq of input waveform
        #3                  ; calculate for remainder of waveform
    endfunc

end
```

The '#' function can also refer to integer variable names, which will be evaluated at run-time to determine which Macro Manager menu input parameter should be used. However, variables used in this way can only be integer, and must evaluate between 1 and 6, inclusive, or else a run-time error will occur.

The '@' function will convert the value of both integer and real variables and constants to string form. Also, string variables constants will be converted by the '@' function. Note that when real variables or constants are converted to string form, the string is trimmed of leading spaces, trailing mantissa zeros, and leading exponent zeros. Making strings as short as possible without altering their value allows them to be used efficiently in the Difference Equations function, and otherwise occupy less space in string manipulation and display operations.

If you are using multiple '#' or '@' functions to refer to variable names in the same data field, you may have to "delimit" the references with parentheses. A Difference Equations example is shown below:

```
integer: doug,jeff,mike

Difference Equations
    y[n]=x1[@doug+n+@jeff]+(@mike)j
    .
    .
    .
endfunc
```

Sometimes parentheses are required for variable names because otherwise it is difficult for the compiler to tell what is variable and what is not. In the example above, note that the first two references, '@doug' and '@jeff', did not require delimiting, because they were followed by characters not allowed in variable names. However, the last reference, '@mike', needed delimit-

ing because it occurred before the imaginary operator j; otherwise the compiler would have searched for a variable named mikej. Below are the rules the compiler uses when parsing variable or constant entry after '#' and '@' function occurrences in data-entry menu fields:

1) Any characters encountered not allowed in variable names (see section 7.2.3, Macro Variables) will terminate the '#' or '@' function parsing, and the characters accumulated up to that point will be used as the variable name or constant value. Also, the end of the macro source line will cause the parsing to terminate.

2) If an integer value (e.g. '1', or '576') occurs immediately after the function, then the entire integer value (until the first non-integer character) is assumed to be part of the function input. If the function input is adjacent to *another* integer, then you must use parentheses, for example:

```
@(jeff2)2*n
```

3) If a variable, after being evaluated and converted to text data at run-time, is to be used to build a menu field input, then parentheses must be used unless the end of the name is also the end of the field, or unless another '#' or '@' reference (or other reserved character that cannot be part of a variable name) occurs immediately after the variable name, for example:

```
name@jeff@doug
```

In the above example, if jeff had a value of 5 and doug a value of 7, then then actual text used would be 'name57'. If the '#' or '@' reference is adjacent to other legitimate variable characters, then you must use parentheses, for example:

```
XX@(jeff)_xx3
```

NOTE: **A maximum of three '#' and/or '@' functions can be used in any one data field.**

7.3.2 Issuing Interactive Commands From Macros

Interactive commands are prefixed with a '$' character to differentiate them from all other macro source constructs. They essentially take the place of the keyboard during Hypersignal functions invoked from macro source files, allowing most data-entry editor, on-line Help, and display commands to be invoked from macro source files.

When invoking an interactive display function from a macro source file, interactive commands are required. Although you could start any function by entering a '$Run' interactive command (which is essentially equivalent to pressing <ctrl-Enter> from the keyboard in manual mode), *a $Run command is required to start all display functions.* Specifying no interactive commands after the data-entry menu fields for a display function will cause the display function to start, but then result in an error condition and possibly hang. Any other initial interactive command other than '$Run' will be interpreted as a data-entry editor command.

As an example, to invoke the Waveform Display/Edit function, you might enter in your macro source file:

```
WAVEFORM DISPLAY/EDIT
    signal              ; input waveform name
    speech              ; dual-waveform display
    3000                ; initial display framesize
    N                   ; no log-amplitude
    1,AD_Units          ; no scale-factor, units
    0                   ; autoscale
    O,C                 ; overlaid display, connected samples
    I                   ; intersected grid

    $Run                ; interactive command:  leave
                        ; data-entry menu, enter display
    $Delay(5000)        ; delay in display 5000 msec (5 sec)
    $Esc                ; return to data-entry menu
    $Esc                ; return to main menu

endfunc
```

Note that any interactive commands inserted before the initial '$Run' or after the first '$Esc' would be interpreted as data-entry editor commands, as shown below with the same example, but modified:

```
WAVEFORM DISPLAY/EDIT
    signal
    speech
    3000
    N
    1,AD_Units
    0
    O,C
    I

    $PgUp               ; data-entry editor pop-up
                        ; directory command
    $Run
    $Delay(5000)
    $Esc
    $PgDn               ; data-entry editor pop-up
                        ; directory next page command
    $Esc

endfunc
```

Although interactive commands are primarily intended for use inside interactive waveform displays, they can also be issued while inside the data-entry editor, as shown in the example above. This can be done either before or after display; however, changing data-entry fields, while possible, should be done carefully, for the following reason. During normal operation, Hypersignal-Macro initializes various fields in data-entry menus automatically based on information contained in waveform headers. This usually happens when a data-entry menu is first entered, requiring the program to load a waveform specified in a waveform filename field. It can also happen when you change waveform filename field entries inside a data-entry menu. This type of operation, while extremely convenient in manual mode, poses a basic conflict with macro operation, since usually one or more data-entry menu fields are specified before the 'endfunc' statements in macro source files. When a function or display is invoked from a macro, all data-entry fields not marked 'no_change' or 'don't_care' (see reserved data-entry menu fields in section 7.3.1 above) take precedence over the automatic load and initialize process. ***However,*** if, from inside the macro file, you use interactive commands to cause ***another*** waveform load (by causing a filename entry field to be entered), then the automatic load will take precedence, and one or more data-entry fields containing previous macro file inputs may be overwritten. Using interactive commands to enter waveform filenames must be done carefully and is only recommended when necessary.

NOTES: **Interactive commands can be processed only during an executing Hypersignal function. More specifically, the following two rules apply:**

1) Interactive commands must start immediately after the data-entry menu fields for the function, and continue uninterrupted until the final '$Esc' sequence or '$Manual' command.

2) An interactive command sequence must end with either the '$Manual' command (see below) one or two '$Esc' commands, depending on the current level of execution (one '$Esc' if ending from the data-entry editor, '$Esc $Esc' if ending a waveform display).

Interactive commands entered outside an executing Hypersignal function (as part of other macro language statements) will result in an error condition.

INTERACTIVE COMMAND REFERENCE

Below is a list of interactive commands. The description for many commands references an equivalent manual mode keyboard or mouse command; for complete information on manual mode equivalents, see sections 3.1, Keyboard and Mouse Interface, and 4.3.1, Interactive Waveform Display Commands.

$Beep

Causes an audible sound.

$Beg

In the data-entry editor, moves cursor to start of current field. In on-line Help display, jumps to start of text. Equivalent to pressing the 'Home' key on the keyboard in manual mode.

$Del

In the Waveform Display/Edit function, causes cut-and-paste delete prompt to appear. Must be followed by '$D' (delete) or '$C' (copy) command to be effective (see section 15.15.1, Waveform Display/Edit Cut & Paste Macro). The display must be in region mode ('$R' command) for this command to operate. Equivalent to pressing 'Delete' key on the keyboard in manual mode.

$Delay(N)

Macro execution is delayed for N msec. N should be of type integer, ranging from 0 to 2,147,483,647 (or about 596 hours).

$DispFrame

In instrumentation displays (Digital Oscilloscope and Spectrum Analzyer), allows the next frame of real-time data to be displayed. For example, a '$Repeat(100) $DispFrame' sequence would allow 100 frames of data to be received from the DSP/Acquisition board and displayed. See also the $Manual interactive command. Has no keyboard or mouse equivalent in manual mode.

$DSPRefresh

Causes DSP board to reset without changing any screen or hardware settings. Equivalent to ctrl-Enter keyboard interactive command

$Down

In waveform displays, makes the target window smaller, or moves the crosshair cursor down. In the data-entry editor, moves the cursor down one field. In on-line Help display, scrolls down one page of text. Equivalent to pressing the down arrow key or moving the mouse down in manual mode.

$DualAutoScale(N)

Enables or disables autoscaling for dual-waveform display (see section 3.3.3.1, Dual Autoscaling). Values for N can be one of the following reserved integer constants: 'Off', 'On', or 'File'. See section 7.2.4, Macro Constants, for the numeric equivalents of these constants). See also the DualAutoScale command below. Please note that when used in its interactive command form above, this command may not take effect until the currently displayed waveforms are reloaded from a data-entry menu.

$End

In the data-entry editor, moves the cursor to the end of the current field. In on-line Help display, jumps to end of text. Equivalent to pressing the 'End' key in manual mode.

$Enter

In waveform displays, causes the display to be redrawn using initial settings. In data-entry editor, enters the current field. In on-line Help display, scrolls down one page of text. Equivalent to pressing the 'Enter' key in manual mode.

$Esc

Causes macro execution to return to the previous menu. Equivalent to pressing the 'Esc' key in manual mode.

$GetDrvParms

Sends message to external device driver asking for updated function/display parameters; used just prior to $RUNINTERACTIVE or $INTERACTIVE commands.

$Help

Invokes on-line Help. Equivalent to pressing the 'F1' key in manual mode.

$Ins

In the Waveform Display/Edit function, causes cut-and-paste insert prompt to appear. Must be followed by '$I' (insert) or '$W' (overwrite) command to be effective (see section 15.15.1, Waveform Display/Edit Cut & Paste Macro). The display must be in region mode ('$R' command) for this command to operate. Equivalent to pressing 'Insert' key on the keyboard in manual mode.

$INTERACTIVE

Place user-controls of current display or instrument in interactive mode.

$Left

In waveform displays, moves the target window or the crosshair cursor to the left. In the data-entry editor, moves the cursor to the left one character. Equivalent to pressing the left arrow key or moving the mouse left in manual mode.

$Manual

Stops macro execution until the current interactive display is *manually exited in full by the user* (by pressing the 'Esc' key to return to the data-entry menu, and the 'Esc' key again to return to the main menu). Before the interactive display function is exited, the user is free to perform all and any number of display operations and commands as usual, as if in manual mode. It may be advisable to invoke a '$Trace(On)' command prior to using this command, so that users will know that the system is in manual mode. If desired, a 'Trace(Off)' command can be invoked after the terminating 'endfunc'.

$PanB

In waveform displays, executes the PREV FRAME interactive display command. Equivalent to pressing the 'Backspace' key on the keyboard in manual mode.

$PanF

In waveform displays, executes the NEXT FRAME interactive display command. Equivalent to pressing the spacebar key on the keyboard or right button on the mouse in manual mode.

$PgDn

In waveform displays, executes the VERTICAL SCALING interactive command (scale down, or make waveforms smaller). In on-line Help display, scrolls down one page of text. In the data-entry editor, displays the next pop-up directory page. Equivalent to pressing the 'PgDn' key on the keyboard in manual mode.

$PgUp

In waveform displays, executes the VERTICAL SCALING interactive command (scale up, or make waveforms smaller). In on-line Help display, scrolls up one page of text. In the data-entry editor, invokes pop-up directory display. Equivalent to pressing the 'PgUp' key on the keyboard in manual mode.

$Print

In waveform displays, executes the built-in PRINT command. Equivalent to pressing the 'F2' key on the keyboard in manual mode.

$Repeat(N)

Repeats the next interactive command N times. N should be of type integer, and must be greater than or equal to zero.

$Refresh

Causes display to refresh without changing any screen or hardware settings. Equivalent ctrl-L keyboard interactive command.

$Right

In waveform displays, moves the target window or the crosshair cursor to the right. In the data-entry editor, moves the cursor one character to the right. Equivalent to pressing the down arrow key or moving the mouse down in manual mode.

$Run

Run display or instrument function. Typically is last command in series of interactive commands.

$SetEscMode

Controls what happens when the Esc key is pressed during macro function exection. A value of zero for N causes Hypersignal to simply abort the executing function when the Esc key is pressed. This has been the traditional Hypersignal operating mode, and is the default mode at the start of all macro functions until the $SetEscMode command is invoked at some point. A value of 1 for N will cause Hypersignal to abort the current function but continue processing the macro. Output data being stored to file, if any, will not be saved. A value of 2 will cause the same action as mode 1, but any output waveform file data being saved or created (e.g. Data acquisition, FFT Generation, etc.) will be written and finalized at the time when the function is aborted. Currently this command is active only for the Analog Conversion function.

$SETFILESTRIPMODE(N)

This command will cause waveform file-stripping to occur (for multichannel waveform acquisition) if N is set to 'ON' (or'1'). N set to 'OFF' or '0'disables file-stripping (default mode). For example:

$SETFILESTRIPMODE(ON)

Waveform file-stripping refers to the process of creating separate ware- form files for each channel in a multichannel waveform file. See section 4.7.3, Using Digital Record/Capture, for more information.

$SETID(xres, yres)

Sets handle of target window when sending information to Windows application.

$SETOFS(xres, yres)

Set screen offset of waveform display following next $RUN command.

$SETTRES(xres, yres)

Sets screen dimensions of waveform display following next $RUN command.

$Trace(N)

Enables or disables interactive command trace display during macro execution. Values for N can be one of the following reserved integer constants: 'Off' or 'On'. (See section 7.2.4, Macro Constants, for the numeric equivalents of these constants). A '$Trace(On)' command causes all interactive commands, as they are executed, to be displayed in the upper left of the waveform display, data-entry menu, or on-line Help display. Trace display is intended to help debug complicated and extensive display command sequences. Also, it may be helpful when creating an automated test procedure in which operators must receive feedback as to the progress of the test sequence. See also the Trace(N) command in section 7.10.2 below.

$Unzoom

In waveform displays, executes the ZOOM-UNDO interactive display command. Equivalent to pressing the two outer mouse buttons in manual mode.

$Up

In waveform displays, makes the target window larger, or moves the crosshair cursor up. In the data-entry editor, moves the cursor up one field. In on-line Help display, scrolls up one page of text. Equivalent to pressing the up arrow key or moving the mouse up in manual mode.

$WaitForDateTime(DT)

Stops macro execution until the system date and time reach the "packed" integer value DT. A date and time can be specified ranging from the year 1980 to the year 2107, with an accuracy of 2 (two) seconds. DT values are calculated as:

(year-1980)*33554432+month*2097152+day*65536+hour*2048+min*32+sec/2

Month should range from 1 to 12. Day should range from 1 to 31. Hour should range from 0 to 23. Minutes and seconds should range from 0 to 59. Note that the multipliers above are equivalent to building the 32-bit DT integer value by shifting the year left by 25 bits, the month by 21 bits, the day by 16 bits, the hour by 11 bits, the minutes by 5 bits, and the seconds *right* by 1 bit.

$WaitForKey(K)

Stops macro execution until the key 'K' is pressed at the keyboard. K is always converted to upper case, and should range from 'A' to 'Z'. It may be advisable to invoke a '$Trace(On)' command prior to using this command, so that users will know that the system is waiting for a key. If desired, a '$Trace(Off)' command can be invoked afterwards.

$Y, INTERACTIVE

This command is intended for use with the Analog Conversion function; it answers a prompt with 'Y' (yes) and causes the function to enter interactive mode at the same time. This allows disk record/playback operations to still run to completion as the macro program , yet remain sensitive to keyboard input during operation.

$Zoom

In waveform displays, executes ZOOM interactive display command. Equivalent to pressing the 'Home' key on the keyboard in manual mode.

NOTE: Currently, the '#' and '@' functions cannot be used in interactive commands that require some type of input. All input to interactive commands that require numeric input must be in the form of numeric constants, and all input to commands that require string input must be in the form of string constants.

OTHER INTERACTIVE ENTRY

Single-character manual mode keyboard commands and multiple-character manual mode entry can be entered as interactive commands by prefixing them with the '$' character. The macro compiler places no restrictions on data entered after the '$' character. For example, to execute a filter design function from a macro source file, you could enter the following interactive command sequence after the data-entry menu fields:

```
$Run            ; leave data-entry menu
$C              ; construct filter
```

As another example, to write a waveform segment to filename "seg_out" from the Waveform Display/Edit function, you could enter the following command sequence:

```
$W              ;  invoke WRITE command
$Seg_out        ; filename input
```

7.4 Macro Input Parameters

The concept of reusable macros requires not only procedures within macros, which makes them callable to other macros, but also input parameters from the outside, which allow the macro to be used for varying input data without changing the macro source file itself. To support this concept, the Macro Manager menu has 6 input parameters that can contain any type of numeric

or string information for use in macro processing. Also, the menu labels for these input parameters can be customized for each macro (see section 4.10.1.2, Parameter Passing From the Macro Manager Menu).

Inside a macro source file, input parameters are referred to as '#1' through '#6'. Also, variables can be used, such as '#parm', where parm is an integer variable that must evaluate between 1 and 6 at run-time. The '#' function reference can also be used with parentheses, as in '#(1)', or '#(parm)', which can be helpful when entering multiple '#' references in a data-entry menu field (see section 7.3.1.1).

NOTE: One advanced way to use the Macro Manager input parameters is to *change them dynamically, during the course of macro execution.* This is done by simply listing a 'Macro Manager' command statement and specifying the new data-entry menu fields as needed during the macro, followed by an 'endfunc' statement. If you do this, be sure to specify a $ESC interactive command prior to the Macro Manager 'endfunc' statement, so that the current macro does not try to run recursively. Subsequent references to input parameters then use the new data. This can be helpful when a macro starts with an initial, user-specified, set of input parameters, but then needs to repeat all or part of itself with new parameters.

Error detection when using '#' function references can be more difficult than in most cases. Most errors possible with this function occur at run-time. This is because the contents of the corresponding Macro Manager menu input parameter field cannot be evaluated until run-time. The only thing that can be checked at compile-time is the 1-6 range requirement when an integer constant is being used as the input, and whether the type is integer when a variable is being used as input.

7.5 Macro Assignment Statements

Hypersignal macro language assignment statements allow new values to be assigned to variables of all types. Variables can be assigned the value of numeric or string constants, the value of other variables, or (with the exception of string variables) the value of an operation between variables. Also, depending on the types of the variables involved, the macro language may make some type-conversions. Below are some examples:

```
X=56789
```

The above example assigns the value of 56789 to X if X is an integer; assigns the value of 56789.0 to X if X is real.

```
X=56789.
X=56789e00
```

The two assignments above are equivalent. In the first case, the decimal point forces X to be of type real, and in the second, the exponent ('E') does the same. If X is not declared as real, an error condition will result.

```
X="the brown dog jumped over the fox"
```

In this case, the use of double-quote marks forces X to be of type string. If not, an error condition will result.

For examples of assignment statements that assign the result of an operation between two variables to a third, see section 7.5.1 below, Macro Operators.

A special case of assignment statements occurs in for-loops. For example, the following loop uses an implied assignment statement:

```
for J=1 to 10                ; for loop: J is assigned values
                             ; from 1 to 10
    statement
    statement
    .
    .
    .
    statement

end
```

Only integer variables can be used in a for-loop statement. Use of real variables will result in an error condition. See section 7.6 below, Macro Loops, for more information on for-loops.

NOTE: **All variables must be declared before use! See section 7.2.3, Macro Variables, for detailed information about declaring variables.**

7.5.1 Macro Operators

Assignment statements can reference a maximum of three variables *or* values and contain a maximum of one operator. Currently, valid operators include addition, subtraction, multiplication, division, and integer division. When an operation involves at least one real variable or constant on the right side of the '=' sign, then the result is assumed to be real. Note that operators must always appear on the right side of the '=' sign.

Some examples of valid assignment statements that involve operators:

A=B+100

In the above statement, if B is real, then the result is assumed to be real and A must be a real variable. If B is integer, then the result is integer and A can be either integer or real (type-conversion is performed if A is real).

A=B+C

In the above statement, if either B or C is real, then the result is assumed to be real and A must be a real variable. If *both* B and C are integer, then the result is integer and A can be either integer or real (automatic type-conversion is performed if A is real). *If both B and C are of type string*, then the two strings will be concatenated. Some other examples of valid string operations:

A=B+"add me"
A=B+#1
A=@B+#1

In the last example above, '#1' is an input parameter from the Macro Manager menu, and B refers to any type of variable. If B is of type integer or real, then '@B' will produce the string equivalent of its current value (see section 7.10.1, Macro Special Functions: '#' and '@'). Note that only the '+' operator allows string variables as operands.

A=B*100

In the above statement, if B is real, then the result is assumed to be real and A must be a real variable. If B is integer, then the result is integer and A can be either integer or real (automatic type-conversion is performed if A is real).

A=B*100.0

In the above statement, B's type doesn't matter, since the numeric constant is real, forcing the result to be real and A to be a real variable.

A=B*C

In the above statement, if either B or C is real, then the result is assumed to be real and A must be a real variable. If *both* B and C are integer, then the result is integer and A can be either integer or real (automatic type-conversion is performed if A is real).

A=B/100
A=B/C

In the above statements, the result is always assumed to be real, regardless of the variable and/or constant types. Thus, A must always be a real variable.

A=B div 100
A=B div C

In the above statements, both variables and/or constants on the right side of the '=' sign are assumed to be integer. The result is always integer; A can be either integer or real (automatic type-conversion is performed if real). Note that spaces are required around the 'div' operator in order to allow it to be differentiated from a variable name.

A=B % 100
A=B % C

In the above statements showing examples of the modulus, or "remainder", operator, both variables and/or constants on the right side of the '=' sign are assumed to be integer. The result is always integer; A can be either integer or real (automatic type-conversion is performed if real). The modulus operator returns the remainder of an integer division; for example, 8 % 3=2, 7 % 7=0, 11 % 1=0, etc.

```
A=B & C
A=B | C
A=B ^ C
A=!B
A=~B
```

The above operators include '&'which operator signifies bitwise AND, '|' which signifies bitwise OR, '^' which signifies bitwise XOR (exclusive OR), ! which signifies "boolean NOT" and which converts a zero value to a one and converts a non-zero value to zero, and '~', which signifies "bitwise NOT" and which performs a 1s-complement operation. In all of the above statements, both variables and/or constants on the right side of the '=' sign are assumed to be integer. The result is always integer; A can be either integer or real (automatic type-conversion is performed if real).

7.6 Macro Loops

Hypersignal macro language loops can contain any number of other macro statements, including other loop constructs. Loops can be of two types: for-loops, and while-loops. Each loop must be terminated by an 'end' statement. Some examples of loop constructs:

```
J=0                         ; while loop
while J < 1
    statement
    statement
    .
    .
    .
    statement
end
for J=1 to 10               ; for loop
    statement
    statement
    .
    .
    .
    statement
end
for A=1 to B                ; for loop
    statement
    statement
    .
    .
    .
    statement
end
```

```
for A=1 to B                    ; nested loop
    statement
    statement
    .
    .
    .
    statement

    while C<>D
        statement
        statement
        .
        .
        .
        statement
    end

end
```

See section 7.5, Macro Assignment Statements, for more information on rules for the assignment portion of for-loops. See section 7.6.1, below, for more information on the comparison portion of while-loops. String variables cannot be used in for-loop or while-loop statements.

7.6.1 Macro Relational Comparisons

Basic comparisons are allowed in Hypersignal macro language for-loop and if-then-else statements:

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
= or **==**	equal to
<> or **!=**	not equal to

The variables on either side of a relational comparison can be both integer, both real, or one of each, in which case the integer variable is type-converted to real prior to comparison. String variables can be used in equal-to and not-equal-to comparisons, but not in any other relational comparisons.

7.7 Macro If-Then-Else Statements

A macro language if-then-else construct can contain any number of other macro statements, including other if-then-else constructs. Each ‘if ... then’ statement (and its associated ‘else’ statement, if applicable) must be terminated by an ‘endif’ statement. Some examples of if-then-else constructs:

```
if x=2 then                     ; if-then
    statement
    statement
    .
    .
     statement

endif
```

```
statement
statement

if x=2 then                    ; if-then-else

    statement
    statement
    .
    .
    statement

else

    statement
    statement
    .
    .
    statement

endif
```

7.8 Macro Procedures

In the Hypersignal macro language, procedures may be defined in order to create re-usable functions that can help make the overall macro more reliable, and also greatly reduce the amount of source code required. *All procedures must be defined prior to the main body of macro source code.*

Integer, real, and string variables may be passed to procedures. Variables can be passed to procedures by reference or by value. Passing by reference means that the procedure is free to change the variable's value, and if so, then the value of the variable outside the procedure will change also. Passing by value means that the procedure should make no attempt to change the variable's value, because a constant could be passed as the variable when the procedure is called. If a constant is passed, and the procedure attempts to change the value of the constant, a run-time error will result. Local variables of all types may also be declared inside the procedure that are "visible" only from within the procedure, and occupy variable space only when the procedure executes. Also, procedures can refer to global variables that are defined in the main body of the macro source file at any time. Note that any global variables (visible from both outside and inside a procedure) needed must be declared *before* the procedure is defined. When writing procedures, local variables are highly recommended whenever possible, because of that amount of space can be saved from the finite global variable space.

Below is an example of a procedure declaration:

```
procedure examp(integer a, real b, real c)
integer temp1
real temp2



    if b>c then
        temp1=1
        temp2=2.5
    else
        temp1=3
        temp2=4.25
    endif

    a=temp1
    b=temp2

return
```

Note that all procedures must be terminated by a 'return' statement. The return statement must be on a separate line, and accepts no parameters. Note also the declaration of local variables in the above example. Unlike declaring global variables elsewhere in a macro source file, procedure variable declarations must have the type first, followed by the variable name. A colon (':') after the type is optional.

Procedures can be called anywhere in the main body of a macro source file (except after a menu display or processing function and before its 'endfunc' statement). Both procedure *calls and declarations* must contain a '()', even if no variables are being passed. Variables being passed in a procedure call should be listed in the order matching their declaration by the procedure definition, and should be separated by commas. No numeric or string constants can be present in a procedure call.

Below is an example of calling the above procedure:

```
examp(x, y, z)
```

In the above example, the integer variable temp1 and the real variable temp2 are used only inside the procedure examp; the values of x and y (but not z) are changed when examp is called.

7.9 Macro Goto Statements

Labels can be defined, and goto statements used which reference the labels, in the macro language. To define a label, simply place its name (or number) followed by a colon either at the beginning of line containing another valid macro statement, or on a separate line. To jump to a label, simply use the word 'goto' and the label name. Below is an example of jumping to a label:

```
lab: x=1
     statement
     statement
     .
     .
     .
     statement
     if a<>b then goto lab
```

7.10 Macro Commands and Functions

The Hypersignal macro language contains several pre-defined commands that perform various functions, including user-interface, scheduling, file-access, and mathematical functions. Below is a list of built-in commands and language commands and functions:

Abort

Causes the currently executing macro to abort completely from Hypersignal without saving the user environment configuration file. Equivalent to pressing a 'QA' sequence from the main menu on the keyboard in manual mode. This command is intended to be used when a macro has been invoked remotely (by another program) by passing the macro source filename in the Hypersignal command line (see section 3.0.1, Advanced Features When Starting Hypersignal). A simple example would be a batch file that included as one of its lines:

```
HSMACRO examp.mac
```

In such situations, the macro source file (in this case, examp.mac) should include as its last command either an 'Abort' or an 'Exit' command (see also the 'Exit' command below).

ret = Abs(var)

Returns |var|. Var can be any integer or real constant or variable. Var can also be a '#' function reference, which should point to a Macro Manager menu field input parameter containing integer or real data at run-time. Ret must be a real variable if var is of type real, or else an error condition will occur at compile-time. If var is of type integer, then ret can be an integer variable.

Beep

Causes an audible sound.

ret = CallDLLFunction(args)

Call the DLL function specified in args, and include function parameters as specified in args. Args should be a string variable with each parameter separated by a space; for example:

args = "SetCursorPos 135 100"

The CallDLLFunction call will convert parameters to numeric values if they are clearly numeric, and treat other parameter as pointers to string values. The result of the DLL function call is returned in ret. Ret can be either a real or integer variable.

ClearDSPMem

Clears the host DSP memory external value write list. Should be executed after a Hypersignal DSP-based function that wrote user-defined values to DSP memory finishes executing, and you wish to return to non-user-defined Hypersignal operation. See the WriteDSPMem function below for more information.

ClearDSPVal

Clears the host DSP memory internal value write list. Should be executed after a Hypersignal DSP-based function that wrote internal values to user-defined DSP memory addresses finishes executing, and you wish to return to non-user-defined Hypersignal operation. See the WriteDSPVal function below for more information.

CloseTextFile(hFile)

The CloseTextFile call closes the currently open text file specified by hFile. hFile should be an integer variable previously set by an OpenTextFile function call.

Delay(N)

Macro execution is delayed for N msec. N can be an integer constant or variable, ranging from 0 to 2,147,483,647 (or about 596 hours).

Display(...text...) or Display(var)

Displays up to 40 characters of the resulting text in the System Status area. Text can be whatever text is entered in the source file inside the parentheses. (Note that a semicolon will be interpreted as the end of displayable text and the start of a comment). Var can be an integer, real, or string variable or constant. In the case of variables, the '@' string convert function (see section 7.10.1 below) will produce the same result as just the variable name. Use of double-quote marks (which indicates a standard string constant) for string data will produce the same result as no quote marks, with one important exception: *string constants cannot contain spaces*. Use of the '#' function reference (see section 7.4) will cause the corresponding Macro Manager menu field input parameter to be displayed.

Currently the Display command initially prints on the top line of the System Status area, and then autoincrements to the next line each time it is invoked. After the System Status area is full, the next Display command will clear the area and start over at the top line again.

Many of the macros provided with the distribution software use this command at the end the macro to indicate that processing has finished. This command is also very useful when the result of macro processing is only one or a few values, which should be displayed instead of written to a waveform file.

DOS Shell

Invokes a DOS shell, which brings up the DOS command line and transfers control to the keyboard. You (or an operator) must type 'Exit' to return to the macro and continue processing.

DualAutoScale(N)

Enables or disables autoscaling for dual-waveform display (see section 3.3.3.1, Dual Autoscaling). Values for N can be one of the following reserved integer constants: 'Off', 'On', or 'File'. N can also be a variable or numeric constant that evaluates to 'Off' or 'On'. See section 7.2.4, Macro Constants, for the numeric equivalents of these constants).

ret = ExecProgram(progname, args, Mode)

Progname should specify the program to execute. Args should specify the command-line parameters required by the program, if any. If there are no command-line parameters, args should be a null string. Mode should be either SYNC (calling macro program waits for completion) or ASYNC (calling macro program continues to run simultaneously). For example, if the calling macro program did this:

ret = ExecProgram(”c:\test.exe”, “/a /b”, SYNC)

The equivalent command-line entry it made by hand would be:

C:\test /a /b

Progname and args should be string variables. Mode should be an integer constant or variable and ret can be either real or integer.

Exit

Causes the currently executing macro to exit completely from Hypersignal after saving the user environment configuration file. Equivalent to pressing an ‘E’ or a ‘QE’ sequence from the main menu on the keyboard in manual mode. This command is intended to be used when a macro has been invoked remotely (by another program) by passing the macro source filename in the Hypersignal command line (see section 3.0.1, Advanced Features When Starting Hypersignal). A simple example would be a batch file that included as one of its lines:

HSMACRO examp.mac

In such situations, the macro source file (in this case, examp.mac) should include as its last command either an ‘Abort’ or an ‘Exit’ command (see also the ‘Abort’ command above).

ret = Exp(var)

Returns e^{var}. Var can be any integer or real constant or variable. Var can also be a ‘#’ function reference, which should point to a Macro Manager menu field input parameter containing integer or real data at run-time. Ret must be a real variable, or else an error condition will occur at compile-time.

ret = GetDataValue(filename, N)

Returns the Nth data value (sample) from the waveform specified by filename. Ret must be an integer or real variable, or else an error condition will occur at compile-time. Filename should be a string variable or constant (any other text is interpreted as a string constant). Automatic type conversion is performed if the assignment variable is real. However, it is recommended that the assignment variable be real because of the small data values (less than one) possible. The filename is searched for on the DRIVE and SUBDIRECTORY specifed in the User Setup area, and should be no more than 12 characters long, including optional filename extension. If no extension is provided, then a “.TIM” extension is assumed. N should specify an integer constant or variable; ‘#’ function references should specify a Macro Manager menu field input parameter containing integer data. N can range from 0 to the waveform length – 1, where N is specified in words (see “word index” in the waveform file data section description in chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT). Several reserved integer constants are available that can be used for N that refer to specific header values; see the GetHeaderValue command description below.

NOTE: When accessing frequency domain magnitude files, the values returned will not be scaled by FRAMESIZE, window factor, or single-sided spectrum factor as explained in section 4.5.3.2, Frequency Domain Display Scaling. If this scaling is needed, you will have to access the appropriate waveform header values, and perform the scaling inside the macro.

ret = GetDateTime()

Returns the current date and time in the form of an integer variable. Ret must be a integer variable, or else an error condition will occur at compile-time.be of type integer. The value returned is “packed”, in the following form:

(Year-1980) * 33554432 + month * 2097152 + day * 65536 + hour * 2048 + min * 32 + sec/2

You can use logical operations and integer division to extract specific values from the packed value. For example, the following code would extract the month and day, place them into integer variables, and then form a string variable that could be used as a filename for further processing:

```
datetime = GetDateTime()
month = datetime & $01E00000
day = datetime & $0001F000
filename = "M" + @month
filename = filename + "D"
filename = filename + @day
```

ret = GetEscape()

Returns 1 if last function was "escaped" or aborted; returns 0 if last function was executed. Ret must be an integer variable, or else an error condition will occur at compile-time.

str = GetExePath()

The GetExePath function returns the original Hypersignal .exe path. Str should be a string variable.

ret = GetHeaderValue(filename, N)

Returns the Nth value from the header of the waveform specified by filename. Ret must be an integer or real variable, or else an error condition will occur at compile-time. Filename should be a string variable or constant (any other text is interpreted as a string constant). Automatic type conversion is performed if the assignment variable is real. However, it is recommended that the assignment variable be integer for all header values except for MAX AMPLITUDE, in which case the assignment variable should be real because of the small values (less than one) possible. The filename is searched for on the DRIVE and SUBDIRECTORY specifed in the User Setup area, and should be no more than 12 characters long, including optional filename extension. If no extension is provided, then a ".TIM" extension is assumed. N should specify an integer constant or variable; '#' function references should specify a Macro Manager menu field input parameter containing integer data. The run-time value for N can range from 0 to 63, where N is specified in words (see "word index" in the waveform file header description in chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT).

Several reserved integer constants are available that can be used for N that refer to specific header values:

Amp	waveform max |amplitude|
Fr	waveform framesize
Fs	waveform sampling frequency
FFTord	waveform FFT order
Len	waveform length
Overlap	waveform overlap amount (samples)
NumChan	waveform number of channels
TypeData	waveform type attribute
Exponent	waveform exponent

ret = GetSysConfig(N)

Returns the string value of the Nth System Configuration menu field. N should range from 0 to 9. Ret must be a string variable, or else an error condition will occur at compile-time.

ret = GetUserSetup(N)

Returns the string value of the Nth User Setup area field. N should range from 0 to 7. Ret must be a string variable, or else an error condition will occur at compile-time.

InitBoard(N)

Initialize a curently installed DSP/data acq. board as specified by the System Config menu field index, N. N can be one of the following values:

0 = board defined in "ANALOG CONVERSION" field

1 = board defined in “DSP ACCELERATOR” FIELD

2 = board defined in “DIGITAL OSCILLOSCOPE” field

3 = board defined in “SPECTRUM ANALYZER” field

For example, to initialize and run the board listed in the DSP Accelerator field in the System Config menu, the following macro language code would be used:

```
InitBoard(1)
```

For more information on specifying board types in the System Config menu, see section 2.5.2, Initializing the System Configuration Menu (Selecting a DSP/Acquisition Board Driver).

ret = Ln(var)

Returns Ln(var). Var can be any integer or real constant or variable. Var can also be a ‘#’ function reference, which should point to a Macro Manager menu field input parameter containing integer or real data at run-time. Ret must be a real variable, or else an error condition will occur at compile-time. Var cannot be less than or equal to zero at run-time, or else an error condition will result.

LoadDSPFile(N, filename)

Load the program file indicated by “filename” to the DSP/data acq. board as specified by the System Config menu field index, N. Filename should be a string variable containing the name of the DSP executable (COFF) file to download. If this parameter is “” (null string), then the default DSP program file is used, or text entry in System Config menu field is used.

N can be one of the values shown in the InitBoard function description.

For example, to initialize the board listed in the DSP Accelerator field in the System Config menu and download the default DSP program file supplied with the Hypersignal-Macro software, the following macro language code would be used

```
InitBoard(1)
LoadDSPFile(1, “ ”)
```

For more information on specifying board types in the System Config menu, see section 2.5.2, Initializing the System Configuration Menu (Selecting a DSP/Acquisition Board Driver).

ret = Log(var)

Returns Log_{10}(var). Var can be any integer or real constant or variable. Var can also be a ‘#’ function reference, which should point to a Macro Manager menu field input parameter containing integer or real data at run-time. Ret must be a real variable, or else an error condition will occur at compile-time. Var cannot be less than or equal to zero at run-time, or else an error condition will result.

ret = NumToHex(var)

Returns the hex representation of var in str. Var can be any integer constant or variable. Ret must be a string variable. ‘0x’ prefixes or ‘h’ suffixes are not added to the string. The result of this function must be assigned to string variable, or else an error condition will occur at compile-time.

hFile = OpenTextFile(filename)

The OpenTextFile call opens the text file specified by filename and returns a file handle, which has value >= zero if the file was opened successfully, and has a negative value if an error occurred. Filename should be a string variable. hFile must be an integer variable, or else an error condition will occur at compile-time.

Below is a short example of reading a text file:

```
integer hFile, flen
string  linestr, dispstr
```

```
hFile = OpenTextFile("filter.fir")          ; open file

SetTextFileMode(1)                          ; prevent macro from aborting on error

if (hFile >= 0) then                        ; valid file handle?
    linestr = ReadTextFile(hFile)
    Display(linestr)                        ; display first line in file

    linestr = ReadTextFile(hFile)
    flen = StrVal(linestr)                  ; convert second line to integer

    if (flen == 102 ) then                  ; check for value

        dispstr = "magic number is " + @flen
        Display(dispstr)
    endif

    CloseTextFile(hFile)                    ; remember to close the file when thru!
endif
```

Currently, Hypersignal macro program execution supports two (2) open text files at one time. This may be increased in the future. Note that text files must be operated on sequentially; there is no seek capability.

ret = PostMessage(args)

Post message to window specified in args, and include wParam and lParam values as specified in args. Args should be a string variable with each parameter separated by a space; for example:

args = "WinClassName 135 100 200"

The PostMessage call will convert the latter three (3) parameters to numeric values, and post the message to the window specified by WinClassName. Note that only the first instance of window with class name WinClassName will receive the message. The result of the PostMessage call is returned in ret. Ret can be either a real or integer variable.

PutDataValue(filename, N, value)

Puts value into the Nth position in the data section of the waveform specified by filename. Filename should be a string variable or constant (any other text is interpreted as a string constant). Value can be integer or real. Filename is searched for on the DRIVE and SUBDIRECTORY specifed in the User Setup area, and should be no more than 12 characters long, including optional filename extension. If no extension is provided, then a ".TIM" extension is assumed. N should specify an integer constant or variable; '#' function references should specify a Macro Manager menu field input parameter containing integer data. The run-time value for N can range from 0 to the waveform length – 1, where N is specified in words (see "word index" in the waveform file data section description in chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT). Several reserved integer constants are available that can be used for N that refer to specific header values; see the GetHeaderValue command description above.

PutHeaderValue(filename, N, value)

Puts value into the Nth position in the header of the waveform specified by filename. Filename should be a string variable or constant (any other text is interpreted as a string constant). Value should be integer, but automatic type conversion is performed if real. Filename is searched for on the DRIVE and SUBDIRECTORY specifed in the User Setup area, and should be no more than 12 characters long, including optional filename extension. If no extension is provided, then a ".TIM" extension is assumed. N should specify an integer constant or variable; '#' function references should specify a Macro Manager menu field input parameter containing integer data. The run-time value for N can range from 0 to 63, where N is specified in words (see "word index" in the waveform file header description in chapter 5, HYPERSIGNAL WAVEFORM FILE FORMAT).

Several reserved integer constants are available that can be used for N that refer to specific header values. See the GetHeaderValue command description above for a complete list of these pre-defined constants.

str = ReadTextFile(hFile)

The ReadTextFile call reads a line from the text file specified by hFile. hFile should be an integer variable previously set by an OpenTextFile function call. The result of the ReadTextFile call is returned in str. Str must be a string variable, or else an error condition will occur at compile-time.

ResetBoard(N)

Reset the board specified by the System Config menu field index , N. The board is held in Reset mode. N can be one of the following values shown in the InitBoard function description.

For example, to initialize, run, and reset the board listed in the DSP Accelerator field in the System Config menu, the following macro language code would be used:

```
InitBoard(1)
LoadDSPFile(1, “”)
RunBoard(1)
ResetBoard(1)
```

ret = Round(var)

Returns var rounded to the nearest integer. Var can be any integer or real constant or variable. Var can also be a ‘#’ function reference, which should point to a Macro Manager menu field input parameter containing integer or real data at run-time. Ret must be an integer variable, or else an error condition will occur at compile-time. Var cannot be less -2147483648.5 or greater than 2147483647.5 at run-time, or else an error condition will result.

RunBoard(N)

Take the board specified by the System Config menu field index N out of reset and start running. N can be one of the values shown in the InitBoard function description.

For example, to initialize and run the board listed in the DSP Accelerator field in the System Config menu, the following macro language code would be used:

```
InitBoard(1)
LoadDSPFile(1, “”)
RunBoard(1)
```

ret = SendMessage(args)

Send message to window specified in args, and include wParam and lParam values as specified in args. Args should be a string variable with each parameter separated by a space; for example:

```
args = “WinClassName 135 100 200”
```

The SendMessage call will convert the latter three (3) parameters to numeric values, and send the message to the window specified by WinClassName. Note that only the first instance of window with class name WinClassName will receive the message. The result of the PostMessage call is returned in ret. Ret can be either a real or integer variable.

SetEscMode(N)

Controls what happens when the Esc key is pressed during macro function exection. A value of zero for N causes Hypersignal to simply abort the executing function when the Esc key is pressed. This has been the traditional Hypersignal operating mode, and is the default mode at the start of all macro functions until the SetEscMode command is invoked at some point. A value of 1 for N will cause Hypersignal to abort the current function but continue processing the macro. Output data being stored to file, if any, will not be saved. A value of 2 will cause the same action as mode 1, but any output waveform file data being saved or created (e.g. Data acquisition, FFT Generation, etc.) will be written and finalized at the time when the function is aborted. Currently this command is active only for the Analog Conversion function.

SetFileStripMode(N)

This command will cause waveform file-stripping to occur (for multichannel waveform acquisition) if N is set to 'ON' (or'1'). N set to 'OFF' or '0'disables file-stripping (default mode). For example:

SetFileStripMode(ON)

Waveform file-stripping refers to the process of creating separate ware- form files for each channel in a multichannel waveform file. See section 4.7.3, Using Digital Record/Capture, for more information.

SetTextFileMode(Modeval)

The SetTextFileMode call sets file mode so text file errors do not cause a macro abort. Modeval should be an integer. A Modeval of 0 given to this command causes default error behavior in waveform file calls (as in previous versions of Hypersignal). A Modeval of 1 specifies that text file errors should not cause a macro abort, and instead return control to the executing macro program. When an error occurs in this mode, text file functions (OpenTextFile, ReadTextFile, etc.) will typically return -1.

SetWaveformFileMode(Modeval)

The SetWavefromFileMode call sets file mode so wavefrom file errors do not cause a macro abort. Modeval should be an integer. A Modeval of 0 given to these commands causes default error behavior in waveform file calls. A Modeval of 1 specifies that waveform file errors should not cause a macro program abort, and instead return control to the executing macro program. When an error occurs in this mode, wareform file functions (GetHeaderValue, PutDataValue, etc.) will typically return -1.

str1 = StrCopy(str2, N1, N2)

Str2 and str1 should be string variables. N1 and N2 should be integers. The StrCopy function copies N2 of characters from str2, starting at position N1, and stores the result in str1. For example:

str1 = StrCopy(str2, 5, 8)

ret = Strlen(str)

The Strlen function returns the length of the string str. Ret can be either real or integer (automatic type conversion is performed if necessary).

ret = StrPos(str1, str2)

Searches str2 (a string variable or constant) for the first occurrence of str1 (a string variable or constant), and returns the position in the string. Ret must be an integer variable, or else an error condition will occur at compile-time. The returned position is "1-based"; that is, for str1 identical to str2, the StrPos function would return a value of 1. The StrPos function returns zero if no occurrence is found.

ret = StrVal(str)

The StrVal function converts string str into a numeric value ret. Ret can be integer or real (automatic type-conversion is performed if necessary).

ret = Sqr(var)

Returns var^2. Var can be any integer or real constant or variable. Var can also be a '#' function reference, which should point to a Macro Manager menu field input parameter containing integer or real data at run-time. Ret must be a real variable if var is of type real, or else an error condition will occur at compile-time. If var is of type integer, then the result can be assigned to an integer variable.

ret = Sqrt(var)

Returns $\sqrt{var}$. Var can be any integer or real constant or variable. Var can also be a '#' function reference, which should point to a Macro Manager menu field input parameter containing integer or real data at run-time. Ret must be a real variable, or else an error condition will occur at compile-time. Var cannot be less zero than at run-time or an error condition will result.

ret = TenX(var)

Returns 10^{var}. Var can be any integer or real constant or variable. Var can also be a '#' function reference, which should point to a Macro Manager menu field input parameter containing integer or real data at run-time. Ret must be a real variable, or else an error condition will occur at compile-time.

Trace(N)

Enables or disables interactive command trace display during macro execution. Values for N can be one of the following reserved integer constants: 'Off' or 'On'. N can also be a variable or numeric constant that evaluates to 'Off' or 'On'. See section 7.3.2, Issuing Interactive Commands From Macros for more detailed information about this command.

ret = Trunc(var)

Returns var truncated to the nearest integer. Var can be any integer or real constant or variable. Var can also be a '#' function reference, which should point to a Macro Manager menu field input parameter containing integer or real data at run-time. Ret must be an integer variable, or else an error condition will occur at compile-time. Var cannot be less or equal to -2147483649 or greater than or equal to 2147483648 at run-time, or else an error condition will result.

ret = UpCase(str)

Returns a string value determined by converting str (a string variable or constant) to all uppercase characters. Non-alpha characters are not altered. Ret must be a string variable, or else an error condition will occur at compile-time.

WaitForDateTime(DT)

Stops macro execution until the system date and time reaches the "packed" integer value DT. A date and time can be specified ranging from the year 1980 to the year 2107, with an accuracy of 2 (two) seconds. DT values are calculated as:

(Year-1980) * 33554432 + month * 2097152 + day * 65536 + hour * 2048 + min * 32 + sec/2

Month should range from 1 to 12. Day should range from 1 to 31. Hour should range from 0 to 23. Minutes and seconds should range from 0 to 59. Note that the multipliers above are equivalent to building the 32-bit DT integer value by shifting the year left by 25 bits, the month by 21 bits, the day by 16 bits, the hour by 11 bits, the minutes by 5 bits, and the seconds *right* by 1 bit.

WaitForKey(K)

Stops macro execution until the key 'K' is pressed at the keyboard. K is always converted to upper case, and should range from 'A' to 'Z'. It may be preferable to use the interactive form of this command ($WaitForKey) while trace display is enabled in order to let users know that the system is waiting for a key.

WriteDSPMem(memtype, addr, val)

Writes val to the DSP memory address in the memory area specified by memtype. Memtype should be a string value of "P" (program memory), "D" (data memory), "X" (data memory for vector data devices), or "Y" (data memory for vector data devices). Addr should be in the range allowed by the DSP device being used. Val can specify either an integer or real value, although real values should only be used for floating-point DSP devices. Some important notes about this function:

1) A full-word access is assumed for all devices. Thus, you may have to adjust the addr value if the DSP allows access to smaller segments of full-words. For example, when writing to a DSP device whose memory can be accessed on byte boundaries, you should use addr values equivalent to the desired byte-address divided by four.

2) This is a delayed function. Values are stored in a host computer "external value list" and not written until you execute a Hypersignal DSP-based function. The maximum length of the list is 16 entries.

3) Real values are converted to the proper DSP chip binary format as they are written.

4) In order to allow the DSP source code that receives values to coexist with the Hypersignal driver communications routines, you should not write data in certain areas of DSP memory. See Appendix A, WriteDSPVal Constants, in the Hardware Reference Guide for information about memory areas that should be avoided for specific DSP devices.

WriteDSPVal(memtype, addr, val)

This routine is similar to the WriteDSPMem routine above except that val signifies an internal Hypersignal value or menu item. This routine can be used to write a wide range of variables used by Hypersignal to control and communicate with DSP/Analog boards to arbitrary memory locations, for use by user-defined DSP source code. In addition to the notes listed above for the WriteDSPMem function, some important items to be aware of about this function follow:

1) A val that specifies a memory address location (for example, a location containing an address used by Hypersignal to download filter coefficients) is treated differently. In this case, addr simply replaces the previously stored memory address (at the location specified by val) at run-time, causing Hypersignal to use the value of addr for transferring data to/from DSP memory. See Appendix A, WriteDSPVal Constants, in the Hardware Reference Guide for more information.

2) This routine should only be used with predefined values of val. Attempting to use this routine for general purpose memory writes is not recommended; WriteDSPMem above should be used for this purpose.

3) This is a delayed function. Values are stored in a host computer "internal value list" and not written until you execute a Hypersignal DSP-based function. The maximum length of the list is 16 entries.

Several reserved integer constants are available that can be used for val that refer to specific DSP variables, as shown below:

DSPBOARD	the type of DSP board in use (see HRG for more explanation)
	starting, analog channel (zero-based)
DSPCHAN	address of filter 1 coefficients
DSPFILTCOEFFADDR1	address of filter 2 coefficients
DSPFILTCOEFFADDR2	length of filter 1 coefficients
DSPFILTLEN1	length of filter 2 coefficients
DSPFILTLEN2	FFT length
DSPFFTLEN	framesize
DSPFR	sampling rate control/mode register value
DSPFSMODE	sampling rate value (in Hz)
DSPFSVALUE	number of analog channels
DSPNUMCHAN	serial port control register value (i.e. AIC, codec, IOC, etc.)
DSPSERIALCTRL	address of FFT output magnitude data
	address of FFT output real/imaginary data
DSPMAGDATAADDR	address of stimulus analog output data
DSPRIDATAADDR	address of time domain input/output data
DSPSTMDATAADDR	address of window data
DSPTIMDATAADDR	
DSPWNDDATAADDR	

For more detailed information about the above constants, and all other values that can be written to arbitrary DSP memory locations, please see Appendix A, WriteDSPVal Constants, in the Hardware Reference Guide.

7.10.1 Macro Special Functions: '#' and '@'

The Hypersignal macro language contains two special functions that are hybrids between an operator and a built-in command. The '#' function refers to overall input parameters to the macro from the outside world currently set in the Macro Manager menu. The '@' function converts the value of variables to string form, for use in string assigment statements, the 'Display' command, or data-entry menu fields.

Partially to make them more like functions, and partially to facilitate use of multiple, or concatenated references on the same line or in the same menu data-entry menu field, the '#' and '@' functions will accept parentheses. For example, assuming that both Macro Manager menu input parameter field #1 and the string variable doug contain the text "waveform", and the integer variable one contains the value 1, all of the following statements are equivalent:

```
val=#1
val=#(one)
val=@(doug)
val=@doug
```

Both of these functions can refer to both user-declared and reserved constants. For example, the statement:

val=@1.2345

would convert the value 1.2345 to a string, and assign this string to the string variable val. In the case of the '#' function however, if a constant is used as input, then it must be integer, and also evaluate between 1 and 6, inclusive. This is because it must reference one of the 6 possible Macro Manager menu input parameters.

Also, both of these functions can refer to variables. The '@' function can refer to either integer or real variables; the '#' function can only refer to integer variables. Of course, if a variable is used as input to the '#' function, then the variable should evaluate between 1 and 6, inclusive, at run-time.

Only the '@' function can refer to string variables. In most cases, using the '@' function with a string variable operates identically to not using the '@' function, since the result is the same string in either case.

The '#' and '@' functions can be especially useful when used in menu data fields when invoking processing and display functions. See section 7.3.1.1, Use of the '#' and '@' Functions in Data Fields, for more information.

7.11 Macro Compiler Commands

7.11.1 Macro Include Files

The Hypersignal macro language supports and encourages the use of include files. These are macro source files (default filename extension of ".MAC") that can be included into the main macro source file at any time. Typically, include files will contain one or more often-used, debugged procedures, allowing them to be invoked conveniently and dependably as new macro source files are created. Include files are expected to be found on the DRIVE and SUBDIRECTORY path specified in the Macro Manager menu, along with the main macro source file.

To tell the macro compiler to begin compiling an include file, specify the include filename with an 'INC' statement, as shown below:

{INC filename.mac}

Include files cannot be nested. Compilation of an include file continues until the entire file has been compiled. Compilation errors indicate the include filename and accurately reflect the line number of the include file. When compilation of an include file finishes, compilation of the main source file resumes with the next line after the 'INC' statement.

7.11.2 Create Text File

{WRITE filename string}

Causes macro language compiler to create a text file and write the string data to it (like the {INC directive, no executable code as a result of the''{}" line itself is produced). If quote marks or other punctuation are included in string, they will also be written to the text file.

7.12 Compiling Macros

Hypersignal contains a built-in macro language compiler. Macro language source files are compiled on-line, after the source filename has been specified in the Macro Manager menu. The compiler can operate in one of three modes, which are also specified in the Macro Manager menu:

1) Always Compile. The specified macro language source file is always compiled, every time the macro is to be run.

2) Make Mode. The specified macro language source file is compiled only if its time & date stamp is later than that of the object file. This mode is intende to eliminate compiling time unless changes have been made to the source file. This is the software default mode.

3) Never Compile. In this mode, the current object file (if found) is always used and the source file is not compiled.

Compiling never occurs until the 'F3' key is pressed from the main menu. At that time, the macro source file specified in the MACRO FILENAME field of the Macro Manager menu is opened for input. All macro source files are expected to be found on the DRIVE and SUBDIRECTORY path specified in the Macro Manager menu. If the source file cannot be found, an error message is displayed. If no filename extension has been entered in the MACRO FILENAME field, a ".MAC" extension is assumed. The resulting object file *always* has an ".MXE" filename extension, and is located on the same path as the source file.

7.13 Executing Macros

Macros are always executed after pressing the 'F3' key from the main menu. At that time, the macro source file specified in the MACRO FILENAME field of the Macro Manager menu is compiled if necessary, and the resulting object file executed. All macro object files are expected to be found on the DRIVE and SUBDIRECTORY path specified in the Macro Manager menu.

Macros can also be executed by specifying the full macro filename (including ".MAC" or other extension) in the Hypersignal command line, which will cause the software to immediately execute the macro. See section 3.0.1, Advanced Features When Starting Hypersignal, for more information about the Hypersignal command line.

Input parameters specified in the Macro Manager menu can be used by a macro file. These parameters are referred to in the macro source file as '#N', where N is the parameter number. In this way, reusable macros can be written that perform a standard function with variable input, and don't require any changes once they've been debugged. Combined with procedures and include files, input parameters provide reusability macro features at all three source levels: procedures, groups of procedures and partial macro files, and complete macro files.

7.14 Macro Errors and Trouble-Shooting

Appendix A contains a list of possible of macro processing errors. Macro error messages are always displayed in the System Status area (upper left of main menu). When a macro processing error occurs, the program immediately returns to manual mode, and the error message remains displayed in the System Status area until a manual command is entered.

When a macro error occurs, you should, *before executing any other keyboard or mouse command:*

1) Write down the error number, for reference into the list below.

2) Note which main menu function was left highlighted.

3) If the error turns out to be a run-time error, try running the offending main menu function manually, to see what error messages are printed that may have flashed by too quickly in the macro, or which data-entry menu field was not valid. If the error occurred during a Filter Construction or Difference Equation function (or any function that maintains a setup file), you should delete the setup file and re-run the macro. This will guarantee that the parameters you see when re-entering the function manually after the error occurs will be the same ones placed there by the macro.

NOTE: **For some functions, after the infamous "error in menu-editor input parameters; macro processing aborted" error message occurs the data-entry menu fields are not saved, and when you go back into the menu manually, they aren't there. This happens because the function did not execute, and thus did not save the menu fields to a setup or save file yet. In this situation, an excellent trouble-shooting technique is as follows: enter a '$Delay(2000)' interactive command just prior to the '$Run' command in the offending function/display call. This will give you time to see what the problem was before the software returns to the main menu. Alternatively, you can replace the delay command with a '$WaitForKey(C) command, which will delay until you press the 'C' key.**

this page intentionally left blank

APPENDIX A: Macro Error Messages

This appendix contains a list of possible of macro processing errors. Macro error messages are always displayed in the System Status area (upper left of main menu). When a macro processing error occurs, the program immediately returns to manual mode, and the error message remains displayed in the System Status area until a manual command is entered. The macro error message list below is divided into four categories:

1) Errors that occurred during a processing or display function during macro execution. Invalid data field entry, analog input hardware problems, and disk file access problems are a few examples.

2) File access errors involving the macro source or object files.

3) Compile-time errors that prevent the macro source file from being compiled. Compile-time errors can easily be differentiated from run-time errors because their display includes the line number of the macro source or include file where the error occurred.

4) Run-time errors, not related to processing or display functions, that occur during macro execution. Invalid variable reference (out of range value), a filename input to a built-in function that can't be found, and a Macro Manager menu field input parameter ("#" function reference) that evaluated to an invalid type or value are a few examples.

PROCESSING AND DISPLAY FUNCTION ERRORS (RUN-TIME)

Error Number	Description
4	Invalid data-entry menu fields caused a processing or display function to attempt to overwrite its own input file.
5	Invalid data field entry caused the FFT Generation function to attempt to apply overlap to an LPC Autocorrelation output waveform.
6	Invalid data field entry caused the FFT Generation function to attempt to apply a non-rectangular window to an LPC Autocorrelation output waveform.
7	One or more data-entry menu fields were invalid and rejected by a processing or display function.
8	A DSP/Acquisition hardware problem occurred during a processing or display function. Make sure the hardware is actually specified in the System Configuration menu. Also, ensure that all memory and I/O base addresses are entered correctly in the System Configuration menu (and the corresponding jumpers and/or switches have been set on the board).
9	A processing or display function using the currently specified DSP/Acquisition hardware was unable to obtain enough memory from the system.
10	A disk access problem occurred during a processing or display function.
11	Not enough disk space was available for output waveform files during a processing or display function.
12	An error occurred reading a filter coefficient file during a processing or display function. There may be a problem with the format of the coefficient file, or the file may be corrupted.
13	A waveform file needed by a processing or display function could not be opened.

MACRO FILE ACCESS ERRORS

Error Number	Description
20	The macro source file could not be located. Make sure the FILENAME, DRIVE, and SUBDIRECTORY fields in the Macro Manager menu are correct. If you have explicitly entered a filename extension in the FILENAME field (default is ".MAC"), make sure it is correct.
21	The macro source file could not be read correctly. There may be a problem with the disk or drive.
22	The macro object file could not be located. Make sure the FILENAME, DRIVE, and SUBDIRECTORY fields in the Macro Manager menu are correct.
23	The macro object file could not be read correctly. There may be a problem with the disk or drive.
24	There is not enough space on the drive and/or subdirectory specified in the Macro Manager menu to store the compiled object file.
25	An error occured while storing the compiled object file. There may be a problem with the disk or drive.
26	You are trying to run a file as a macro that is not a Hypersignal-generated macro object file.

MACRO COMPILE-TIME ERRORS

Error Number	Description
30	The indicated source line contains less than three (3) parameters in a loop statement. One or more of the following was missing: variable name, starting value, ending value.
31	The indicated source line contains less than three (3) parameters in an assignment statement. One or more of the following was missing: destination variable name, source variable name, "=" sign.
32	The indicated source line contains a real constant or real variable argument to a built-in function that requires an integer argument.
33	More data-entry menu fields were entered than allowed by the processing or display function that was being invoked. Make sure you have specified the "endfunc" command at the end of all data-entry menu fields and, if any, all interactive commands.
34	The indicated source line attempted to alter a constant. The constant is on the lefthand side of an "=" sign.
35	The indicated source line attempted to alter a function. The function is on the lefthand side of an "=" sign.
36	The indicated source line contains an incorrectly specified "#" function reference to a Macro Manager menu field input parameter. The input parameter specification was to a null string.
37	The indicated source line contains a mismatched procedure call. The number or type of parameters mismatches the previous procedure declaration.
38	The indicated source line contains a parameter in a procedure call that cannot be referenced. Probably an undeclared variable, or an invalid constant.
39	The indicated source line contains an incorrectly specified "#" function reference to a Macro Manager menu field input parameter. The input parameter specification was to an integer value that was out of range (either less than 1 or greater than 6).

40 The indicated source line contains an invalid or unrecognized operator in an assignment statement.

41 The macro source file contains too many variables. Try re-using temporary variables, or placing repetitive blocks of source code into procedures.

42 The indicated source line contains an invalid or unrecognized operand in an assignment, loop, or if-then-else statement.

43 Too many “end” statements were found to properly end all loop or nested loop constructs.

44 Too few “end” statements were found to properly end all loop or nested loop constructs.

45 The indicated source line is totally unrecognized by the macro compiler. The line may contain “hidden” characters that your text-editor doesn’t display.

46 The indicated source line contains an incorrectly specified “#” function reference to a Macro Manager menu field input parameter. The input parameter specification was to a real value (input parameters must be integer values in the range from 1 to 6, inclusive).

47 The indicated source line contains an invalid if-then-else statement.

48 Too many “endif” statements were found to properly end all if-then-else or nested if-then-else constructs.

49 The macro source file contains a mismatched “else” statement.

50 The indicated source line is attempting to define a label that has already been defined.

51 The macro source file attempted to execute a “goto” statement to a label that was never defined.

52 The indicated source line is longer than 80 characters.

53 The indicated source line is attempting to declare a procedure name that has already been declared.

54 The indicated source line contains an invalid procedure declaration.

55 The indicated source line is attempting to call an undefined procedure.

56 The indicated source line did not correctly invoke the built-in command or function.

57 The indicated source line contains an invalid waveform filename in a built-in function call. Filename arguments to function calls must be string constants or variables, or “#” or “@” function references.

58 The indicated source line contains an invalid parameter in a built-in function or command. Function parameters can be restricted both by variable or constant type and, if a constant, by value. See the individual function descriptions in section 7.10 for specific information.

59 The indicated source line contains an “@” reference to an invalid variable or constant. The reference was either null or to a variable that is undeclared, or to an invalid constant.

60 The indicated source line contains an incorrectly specified “#” function reference to a Macro Manager menu field input parameter. The input parameter specification was either to a non-integer variable or constant, or if an integer constant, was not in the range of 1 to 6, inclusive.

61 The indicated source line contains an invalid include file reference. Correct include file reference is of the form:

{INC filename.MAC}

62 The indicated source line is attempting to define a nested procedure, which are not allowed.

63 The indicated source line is attempting to declare a variable name that has already been declared.

64 The indicated source line contains an invalid variable declaration format. See section 7.2.3, Macro Variables.

65 The indicated source line is declaring a variable that has an invalid name (probably contains an illegal character). See section 7.2.3, Macro Variables.

66 The indicated source line is referencing an undeclared variable, undefined function name, or undefined procedure name.

67 The indicated source line is attempting to declare a variable name that is reserved by the macro language as a pre-defined constant. See section 7.2.4, Macro Constants, for a list of reserved names.

68 The indicated source line is attempting to perform a string operation that is not allowed.

69 The indicated source line is attempting to assign variable or constant to another variable of the wrong type, or is attempting to operate on one or more variables of the wrong type. Assignment mismatches include such cases as assigning a string constant or variable to a numeric variable, reference to the "#" and "@" functions on the left side of the "=" sign, and other situations.

MACRO RUN-TIME ERRORS

Error Number	Description
70	An unknown command was found in the macro object file. This is a serious error, indicating that the macro object file may be corrupted, or the macro portion of the distribution software may be corrupted. Try placing the Macro Manager menu COMPILER MODE field in the "compile always" mode ('C' entry). If you still receive this error, please contact your distributor.
71	Macro execution was aborted from the keyboard by the 'F3' key.
72	Run-time version of compile-time error 33. More data-entry menu fields were entered than allowed by the processing or display function that was being invoked. Make sure you have specified the "endfunc" command at the end of all data-entry menu fields and, if any, all interactive commands.
74	A "#" function reference to a Macro Manager input parameter evaluated to an invalid or out-of-range value at run-time.
75	A run-time assignment mismatch occurred due to a Macro Manager input parameter that was undetermined at compile-time.
76	A waveform filename input to a built-in command or function could not be located on the User Setup path at run-time. The path for all waveform filenames is assumed to be the DRIVE and SUBDIRECTORY field values in the User Setup area of the main menu.
79	A procedure attempted to alter a parameter "passed by value", such as a constant. Only parameters "passed by reference" (i.e. global variables in the procedure call) can be altered by the actual procedure code.
80	A divide-by-zero error occurred at run-time.

81	Only addition (concatenation) of string variables allowed; currently this condition can only be detected at run-time.
82	A built-in command or function parameter (not “#” or “@” functions) referenced a variable that evaluated to an invalid value at run-time.
84	Input to math function was less than or equal to zero at run-time.
85	Input to math function was less than zero at run-time.

this page intentionally left blank

APPENDIX B: List of Filename Extensions

This appendix contains a list of all filename extensions created at one time or another by Hypersignal software. Other filename extensions are used or created by the software, including executable DSP code files (such as .OUT) and DSP source code files (such as .ASM), but they are not listed here because they are not unique to Hypersignal.

Exten	File Type	Description
.ACQ	ASCII	File Acquisition function setup file; contains data-entry menu field values associated with the waveform specified by the FILENAME field; see section 4.1.1
.ADC	ASCII	Analog Conversion function setup file; contains data-entry menu field values associated with the waveform specified by the FILENAME field; see section 4.1.2
.BAK	binary	backup waveform file created by Waveform Display/Edit function when a waveform has been edited; see section 4.3.4
.BZZ	binary	temporary intermediate backup waveform file created by Waveform Display/Edit function when a waveform has been edited; see section 4.3.4
.CNF	ASCII	Environment-save configuration file; see section 3.7
.EQU	ASCII	Difference Equations function setup file; contains data-entry menu field values associated with the waveform specified by the FILENAME field
.FFD	ASCII	FIR filter data-entry menu field values and design-goal frequency response associated with the waveform specified by the FILENAME field; see section 4.8.3.2
.FIR	ASCII	FIR filter coefficient file; see section 4.8.3.1
.FRQ	binary	frequency domain magnitude waveform file; see chapter 5
.IFD	ASCII	IIR filter data-entry menu field values and design-goal frequency response associated with the waveform specified by the FILENAME field; see section 4.8.3.2
.IIR	ASCII	IIR filter coefficient file; see section 4.8.3.1
.IMA	binary	frequency domain imaginary component waveform file; see chapter 5
.MAC	ASCII	macro language source file; see section 7.1
.MCF	ASCII	Macro Manger menu configuration file; see section 7.1
.MXE	binary	macro language executable file; see section 7.1
.NUL	binary	temporary file created by Print Display function when testing for printer availability; see section 4.3.2
.PHZ	binary	frequency domain phase waveform file; see chapter 5
.PWR	binary	frequency domain averaged magnitude waveform file; see chapter 5
.REA	binary	frequency domain real component waveform file; see chapter 5
.TIM	binary	time domain data waveform file; see chapter 5
.TMP	binary	temporary intermediate waveform files created by Waveform Display/Edit function during waveform editing; see section 4.3.4
_X.TXT	ASCII	binary-to-ASCII waveform export file, where “X” can be T for time domain, F for frequency domain magnitude, P for frequency domain phase, R for frequency domain real component, I for frequency domain imaginary component, or W for frequency domain power spectra; see section 4.1.4

this page intentionally left blank